Package ‘mFilter’

February 20, 2015

Title Miscellaneous time series filters
Date 2007-10-2
Version 0.1-3
Author Mehmet Balciar <mbalcilar@yahoo.com>
Depends R (>= 2.2.0), stats
Suggests tseries, pastecs, locfit, tseriesChaos, RTisean, tsDyn, forecast
Description The package implements several time series filters useful
for smoothing and extracting trend and cyclical components of a
time series. The routines are commonly used in economics and
finance, however they should also be interest to other areas.
Currently, Christiano-Fitzgerald, Baxter-King,
Hodrick-Prescott, Butterworth, and trigonometric regression
filters are included in the package.
Maintainer Mehmet Balciar <mbalcilar@yahoo.com>
License GPL (>= 2)
URL http://www.mbalcilar.net/mFilter, http://www.r-project.org
Repository CRAN
Date/Publication 2007-11-06 10:00:46
NeedsCompilation no

R topics documented:

  mFilter-package ....................................................... 2
  bkfilter .......................................................... 5
  bwfilter ........................................................... 8
  cffilter ............................................................ 11
  hpfilt er ........................................................... 14
  mFilter ........................................................... 16
  mFilter-methods ..................................................... 19
  trfilter ........................................................... 21
  unemp .............................................................. 23
Description

Getting started with the mFilter package

Details

This package provides some tools for decomposing time series into trend (smooth) and cyclical (irregular) components. The package implements some commonly used filters such as the Hodrick-Prescott, Baxter-King and Christiano-Fitzgerald filter.

For loading the package, type:

```r
library(mfilter)
```

A good place to start learning the package usage is to examine examples for the `mfilter` function. At the R prompt, write:

```r
example("mfilter")
```

For a full list of functions exported by the package, type:

```r
ls("package:mfilter")
```

Each exported function has a corresponding man page (some man pages are common to more functions). Display it by typing

```r
help(functionName).
```

Almost all filters in this package can be put into the following framework. Given a time series \( \{x_t\}_{t=1}^T \) we are interested in isolating component of \( x_t \), denoted \( y_t \), with period of oscillations between \( p_l \) and \( p_u \), where \( 2 \leq p_l < p_u < \infty \).

Consider the following decomposition of the time series

\[
x_t = y_t + \bar{x}_t
\]

The component \( y_t \) is assumed to have power only in the frequencies in the interval \( \{(a, b) \cup (-a, -b)\} \in (-\pi, \pi) \). \( a \) and \( b \) are related to \( p_l \) and \( p_u \) by

\[
a = \frac{2\pi}{p_u} \quad b = \frac{2\pi}{p_l}
\]

If infinite amount of data is available, then we can use the ideal bandpass filter

\[
y_t = B(L)x_t
\]

where the filter, \( B(L) \), is given in terms of the lag operator \( L \) and defined as

\[
B(L) = \sum_{j=-\infty}^{\infty} B_j L^j, \quad L^k x_t = x_{t-k}
\]
The ideal bandpass filter weights are given by

\[ B_j = \frac{\sin(jb) - \sin(ja)}{\pi j} \]

\[ B_0 = \frac{b - a}{\pi} \]

The finite sample approximation to the ideal bandpass filter uses the alternative filter

\[ y_t = \hat{B}(L)x_t = \sum_{j=-n}^{n} \hat{B}_{t,j}x_{t+j} \]

Here the weights, \( \hat{B}_{t,j} \), of the approximation is a solution to

\[ \hat{B}_{t,j} = \arg \min E\{(y_t - \hat{y}_t)^2\} \]

The Christiano-Fitzgerald filter is a finite data approximation to the ideal bandpass filter and minimizes the mean squared error defined in the above equation.

Several band-pass approximation strategies can be selected in the function cffilter. The default setting of cffilter returns the filtered data \( \hat{y}_t \) associated with the unrestricted optimal filter assuming no unit root, no drift and an iid filter.

If theta is not equal to 1 the series is assumed to follow a moving average process. The moving average weights are given by theta. The default is theta=1 (iid series). If theta= \( (\theta_1, \theta_2, \ldots) \) then the series is assumed to be

\[ x_t = \mu + 1_{\text{root}}x_{t-1} + \theta_1 e_t + \theta_2 e_{t-1} + \ldots \]

where \( 1_{\text{root}} = 1 \) if the option root=1 and \( 1_{\text{root}} = 0 \) if the option root=0, and \( e_t \) is a white noise.

The Baxter-King filter is a finite data approximation to the ideal bandpass filter with following moving average weights

\[ y_t = \hat{B}(L)x_t = \sum_{j=-n}^{n} \hat{B}_jx_{t+j} = \hat{B}_0x_t + \sum_{j=1}^{n} \hat{B}_j(x_{t-j} + x_{t+j}) \]

where

\[ \hat{B}_j = B_j - \frac{1}{2n+1} \sum_{j=-n}^{n} B_j \]

The Hodrick-Prescott filter obtains the filter weights \( \hat{B}_j \) as a solution to

\[ \hat{B}_j = \arg \min E\{(y_t - \hat{y}_t)^2\} = \arg \min \left\{ \sum_{t=1}^{T}(y_t - \hat{y}_t)^2 + \lambda \sum_{t=2}^{T-1}(\hat{y}_{t+1} - 2\hat{y}_t + \hat{y}_{t-1})^2 \right\} \]

The Hodrick-Prescott filter is a finite data approximation with following moving average weights

\[ \hat{B}_j = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{4\lambda(1 - \cos(\omega))^2}{1 + 4\lambda(1 - \cos(\omega))^2} e^{i\omega j} d\omega \]
The digital version of the Butterworth highpass filter is described by the rational polynomial expression (the filter’s z-transform)

\[
\frac{\lambda(1 - z^n)(1 - z^{-1})^n}{(1 + z^n(1 + z^{-1})^n + \lambda(1 - z^n(1 - z^{-1})^n)}
\]

The time domain version can be obtained by substituting \(z\) for the lag operator \(L\).

Pollock (2000) derives a specialized finite-sample version of the Butterworth filter on the basis of signal extraction theory. Let \(s_t\) be the trend and \(c_t\) cyclical component of \(y_t\), then these components are extracted as

\[
y_t = s_t + c_t = \frac{(1 + L)^n}{(1 - L)^d} \nu_t + (1 - L)^{n-d} \varepsilon_t
\]

where \(\nu_t \sim N(0, \sigma^2_\nu)\) and \(\varepsilon_t \sim N(0, \sigma^2_\varepsilon)\).

Let \(T\) be even and define \(n_1 = T/p_u\) and \(n_2 = T/p_l\). The trigonometric regression filter is based on the following relation

\[
y_t = \sum_{j=n_2}^{n_1} \{a_j \cos(\omega_j t) + b_j \sin(\omega_j t)\}
\]

where \(a_j\) and \(b_j\) are the coefficients obtained by regressing \(x_t\) on the indicated sine and cosine functions. Specifically,

\[
a_j = \frac{T}{2} \sum_{t=1}^{T} \cos(\omega_j t) x_t, \quad \text{for } j = 1, \ldots, T/2 - 1
\]

\[
a_j = \frac{T}{2} \sum_{t=1}^{T} \cos(\omega_j t) x_t, \quad \text{for } j = T/2
\]

and

\[
b_j = \frac{T}{2} \sum_{t=1}^{T} \sin(\omega_j t) x_t, \quad \text{for } j = 1, \ldots, T/2 - 1
\]

\[
b_j = \frac{T}{2} \sum_{t=1}^{T} \sin(\omega_j t) x_t, \quad \text{for } j = T/2
\]

Let \(\hat{B}(L)x_t\) be the trigonometric regression filter. It can be showed that \(\hat{B}(1) = 0\), so that \(\hat{B}(L)\) has a unit root for \(t = 1, 2, \ldots, T\). Also, when \(\hat{B}(L)\) is symmetric, it has a second unit root in the middle of the data for \(t\). Therefore it is important to drift adjust data before it is filtered with a trigonometric regression filter.

If \(\text{drift}=\text{TRUE}\) the drift adjusted series is obtained as

\[
\tilde{x}_t = x_t - t \left( \frac{x_T - x_1}{T - 1} \right), \quad t = 0, 1, \ldots, T - 1
\]

where \(\tilde{x}_t\) is the undrifted series.

Author(s)

Mehmet Balcilar, <mbalcilar@yahoo.com>

References


### See Also

*mFilter-methods* for listing all currently available *mFilter* methods. For help on common interface function "mFilter", *mFilter*. For individual filter function usage, *bwfilter, bkfilter, cffilter, hpfilter, trfilter*.

---

**bkfilter**

*Baxter-King filter of a time series*

---

**Description**

This function implements the Baxter-King approximation to the band pass filter for a time series. The function computes cyclical and trend components of the time series using band-pass approximation for fixed and variable length filters.

**Usage**

```r
bkfilter(x, pl=NULL, pu=NULL, nfix=NULL, type=c("fixed","variable"), drift=FALSE)
```

**Arguments**

- `x` : a regular time series
- `type` : character, indicating the filter type, "fixed", for the fixed length Baxter-King filter (default), "variable", for the variable length Baxter-King filter.
- `pl` : integer. minimum period of oscillation of desired component (pl<=2).
- `pu` : integer. maximum period of oscillation of desired component (2<=pl<pu<infinity).
- `drift` : logical, FALSE if no drift in time series (default), TRUE if drift in time series.
- `nfix` : sets fixed lead/lag length or order of the filter. The *nfix* option sets the order of the filter by 2*<em>nfix</em>+1. The default is frequency(*x*)*3.
Details

Almost all filters in this package can be put into the following framework. Given a time series \( \{x_t\}_{t=1}^T \) we are interested in isolating component of \( x_t \), denoted \( y_t \), with period of oscillations between \( p_l \) and \( p_u \), where \( 2 \leq p_l < p_u < \infty \).

Consider the following decomposition of the time series

\[ x_t = y_t + \tilde{x}_t \]

The component \( y_t \) is assumed to have power only in the frequencies in the interval \( \{(a, b) \cup (-a, -b)\} \in (-\pi, \pi) \). \( a \) and \( b \) are related to \( p_l \) and \( p_u \) by

\[ a = \frac{2\pi}{p_u}, \quad b = \frac{2\pi}{p_l} \]

If infinite amount of data is available, then we can use the ideal bandpass filter

\[ y_t = B(L) x_t \]

where the filter, \( B(L) \), is given in terms of the lag operator \( L \) and defined as

\[ B(L) = \sum_{j=-\infty}^{\infty} B_j L^j, \quad L^k x_t = x_{t-k} \]

The ideal bandpass filter weights are given by

\[ B_j = \frac{\sin(jb) - \sin(ja)}{\pi j}, \quad B_0 = \frac{b-a}{\pi} \]

The Baxter-King filter is a finite data approximation to the ideal bandpass filter with following moving average weights

\[ y_t = \hat{B}(L) x_t = \sum_{j=-n}^{n} \hat{B}_j x_{t+j} = \hat{B}_0 x_t + \sum_{j=1}^{n} \hat{B}_j (x_{t-j} + x_{t+j}) \]

where

\[ \hat{B}_j = B_j - \frac{1}{2n+1} \sum_{j=-n}^{n} B_j \]

If \( \text{drift=} \text{TRUE} \) the drift adjusted series is obtained

\[ \tilde{x}_t = x_t - t \left( \frac{x_T - x_1}{T-1} \right), \quad t = 0, 1, \ldots, T-1 \]

where \( \tilde{x}_t \) is the undrifted series.

Value

A "mFilter" object (see mFilter).
Author(s)

Mehmet Balcilar, <mbalcilar@yahoo.com>

References


See Also

mFilter, bwfilter, cffilter, hpfilter, trfilter

Examples

```r
## library(mFilter)

data(unemp)

opar <- par(no.readonly=TRUE)

unemp.bk <- bkfilter(unemp)
plot(unemp.bk)
unemp.bk1 <- bkfilter(unemp, drift=TRUE)
unemp.bk2 <- bkfilter(unemp, pl=8, pu=40, drift=TRUE)
unemp.bk3 <- bkfilter(unemp, pl=2, pu=60, drift=TRUE)
unemp.bk4 <- bkfilter(unemp, pl=2, pu=40, drift=TRUE)

par(mfrow=c(2,1), mar=c(3,3,2,1), cex=.8)
plot(unemp.bk1$y, 
    main="Baxter-King filter of unemployment: Trend, drift=TRUE", 
    col=1, ylab="")
lines(unemp.bk1$trend, col=2)
lines(unemp.bk2$trend, col=3)
lines(unemp.bk3$trend, col=4)
lines(unemp.bk4$trend, col=5)
legend("topleft", legend=c("series", "pl=2, pu=32", "pl=8, pu=40", 
                        "pl=2, pu=60", "pl=2, pu=40"), col=1:5, lty=rep(1,5), ncol=1)

plot(unemp.bk1$y, 
    main="Baxter-King filter of unemployment: Cycle, drift=TRUE", 
    col=1, ylab="")
lines(unemp.bk1$cycle, col=2)
lines(unemp.bk2$cycle, col=3)
```

bwfilter

*Butterworth filter of a time series*

**Description**

Filters a time series using the Butterworth square-wave highpass filter described in Pollock (2000).

**Usage**

```r
bwfilter(x, freq=NULL, nfix=NULL, drift=FALSE)
```

**Arguments**

- `x`: a regular time series
- `nfix`: sets the order of the filter. The default is `nfix=2`, when `nfix=NULL`.
- `freq`: integer, the cut-off frequency of the Butterworth filter. The default is `trunc(2.5*frequency(x))`.
- `drift`: logical, `FALSE` if no drift in time series (default), `TRUE` if drift in time series.

**Details**

Almost all filters in this package can be put into the following framework. Given a time series \( \{x_t\}_{t=1} \) we are interested in isolating component of \( x_t \), denoted \( y_t \) with period of oscillations between \( p_l \) and \( p_u \), where \( 2 \leq p_l < p_u < \infty \).

Consider the following decomposition of the time series

\[
x_t = y_t + \bar{x}_t
\]

The component \( y_t \) is assumed to have power only in the frequencies in the interval \( \{(a, b) \cup (-a, -b)\} \subset (-\pi, \pi) \). \( a \) and \( b \) are related to \( p_l \) and \( p_u \) by

\[
a = \frac{2\pi}{p_u} \quad b = \frac{2\pi}{p_l}
\]

If infinite amount of data is available, then we can use the ideal bandpass filter

\[
y_t = B(L)x_t
\]
where the filter, $B(L)$, is given in terms of the lag operator $L$ and defined as

$$B(L) = \sum_{j=-\infty}^{\infty} B_j L^j, \quad L^k x_t = x_{t-k}$$

The ideal bandpass filter weights are given by

$$B_j = \frac{\sin(jb) - \sin(ja)}{\pi j}$$

$$B_0 = \frac{b - a}{\pi}$$

The digital version of the Butterworth highpass filter is described by the rational polynomial expression (the filter’s z-transform)

$$\frac{\lambda(1-z)^n(1-z^{-1})^n}{(1+z)^n(1+z^{-1})^n + \lambda(1-z)^n(1-z^{-1})^n}$$

The time domain version can be obtained by substituting $z$ for the lag operator $L$.

Pollock derives a specialized finite-sample version of the Butterworth filter on the basis of signal extraction theory. Let $s_t$ be the trend and $c_t$ cyclical component of $y_t$, then these components are extracted as

$$y_t = s_t + c_t = \frac{(1 + L)^n}{(1 - L)^n} \nu_t + \frac{(1 - L)^n}{(1 - L)^n} \varepsilon_t$$

where $\nu_t \sim N(0, \sigma_{\nu}^2)$ and $\varepsilon_t \sim N(0, \sigma_{\varepsilon}^2)$.

If drift=TRUE the drift adjusted series is obtained as

$$\tilde{x}_t = x_t - t \left( \frac{x_T - x_1}{T-1} \right), \quad t = 0, 1, \ldots, T - 1$$

where $\tilde{x}_t$ is the undrifted series.

**Value**

A "mFilter" object (see mFilter).

**Author(s)**

Mehmet Balcilar, <mbalcilar@yahoo.com>

**References**


See Also

\[ \texttt{mfilter, hpfilter, cffilter, bkfilter, trfilter} \]

Examples

```r
## library(mFilter)
data(unemp)
opar <- par(no.readonly=TRUE)
unemp.bw <- bwfilter(unemp)
plot(unemp.bw)
unemp.bw1 <- bwfilter(unemp, drift=TRUE)
unemp.bw2 <- bwfilter(unemp, freq=8, drift=TRUE)
unemp.bw3 <- bwfilter(unemp, freq=10, nfix=3, drift=TRUE)
unemp.bw4 <- bwfilter(unemp, freq=10, nfix=4, drift=TRUE)

par(mfrow=c(2,1),mar=c(3,3,2,1),cex=.8)
plot(unemp.bw1$x, main="Butterworth filter of unemployment: Trend, drift=TRUE", col=1, ylab="")
lines(unemp.bw1$trend, col=2)
lines(unemp.bw2$trend, col=3)
lines(unemp.bw3$trend, col=4)
lines(unemp.bw4$trend, col=5)
legend("topleft", legend=c("series", "freq=10, nfix=2", "freq=8, nfix=2", "freq=10, nfix=3", "freq=10, nfix=4"), col=1:5, lty=rep(1,5), ncol=1)

plot(unemp.bw1$cycle, main="Butterworth filter of unemployment: Cycle, drift=TRUE", col=2, ylab="", ylim=range(unemp.bw3$cycle, na.rm=TRUE))
lines(unemp.bw2$cycle, col=3)
lines(unemp.bw3$cycle, col=4)
lines(unemp.bw4$cycle, col=5)
## legend("topleft", legend=c("series", "freq=10, nfix=2", "freq=8, nfix=2", "freq=10, nfix=3", "freq=10, nfix=4"), col=1:5, lty=rep(1,5), ncol=1)
par(opar)
```
**cffilter**

Christiano-Fitzgerald filter of a time series

**Description**

This function implements the Christiano-Fitzgerald approximation to the ideal band pass filter for a time series. The function computes cyclical and trend components of the time series using several band-pass approximation strategies.

**Usage**

```r
cffilter(x, pl=NULL, pu=NULL, root=FALSE, drift=FALSE, 
        type=c("asymmetric","symmetric","fixed","baxter-king","trigonometric"), 
        nfix=NULL, theta=1)
```

**Arguments**

- `x`: a regular time series.
- `pl`: minimum period of oscillation of desired component (pl<=2).
- `pu`: maximum period of oscillation of desired component (2<=pl<pu<infinity).
- `root`: logical, FALSE if no unit root in time series (default), TRUE if unit root in time series. The root option has no effect if type is "baxter-king" or "trigonometric".
- `drift`: logical, FALSE if no drift in time series (default), TRUE if drift in time series.
- `nfix`: sets fixed lead/lag length or order of the filter with "baxter-king" and "fixed". The nfix option sets the order of the filter by 2*nfix+1. The default is nfix=1.
- `theta`: moving average coefficients for time series model: \( x(t) = \mu + \text{root} \times x(t-1) + \theta(1) \times e(t) + \theta(2) \times e(t-1) + \ldots \), where e(t) is a white noise.

**Details**

Almost all filters in this package can be put into the following framework. Given a time series \( \{x_t\}_{t=1}^{T} \) we are interested in isolating component of \( x_t \), denoted \( y_t \) with period of oscillations between \( p_l \) and \( p_u \), where \( 2 \leq p_l < p_u < \infty \).

Consider the following decomposition of the time series

\[
x_t = y_t + \bar{x}_t
\]

The component \( y_t \) is assumed to have power only in the frequencies in the interval \( \{(a,b) \cup (-a,-b)\} \in (-\pi, \pi) \). a and b are related to \( p_l \) and \( p_u \) by

\[
a = \frac{2\pi}{p_u} \quad b = \frac{2\pi}{p_l}
\]
If infinite amount of data is available, then we can use the ideal bandpass filter

\[ y_t = B(L)x_t \]

where the filter, \( B(L) \), is given in terms of the lag operator \( L \) and defined as

\[ B(L) = \sum_{j=-\infty}^{\infty} B_j L^j, \quad L^k x_t = x_{t-k} \]

The ideal bandpass filter weights are given by

\[ B_j = \frac{\sin(jb) - \sin(ja)}{\pi j} \]

\[ B_0 = \frac{b - a}{\pi} \]

The finite sample approximation to the ideal bandpass filter uses the alternative filter

\[ y_t = \hat{B}(L)x_t = \sum_{j=-n_1}^{n_2} \hat{B}_{t,j} x_{t+j} \]

Here the weights, \( \hat{B}_{t,j} \), of the approximation is a solution to

\[ \hat{B}_{t,j} = \text{arg min} E\{(y_t - \hat{y}_t)^2\} \]

The Christiano-Fitzgerald filter is a finite data approximation to the ideal bandpass filter and minimizes the mean squared error defined in the above equation.

Several band-pass approximation strategies can be selected in the function `cffilter`. The default setting of `cffilter` returns the filtered data \( \hat{y}_t \) associated with the unrestricted optimal filter assuming no unit root, no drift and an iid filter.

If \( \theta \) is not equal to 1 the series is assumed to follow a moving average process. The moving average weights are given by \( \theta \). The default is \( \theta=1 \) (iid series). If \( \theta= (\theta_1, \theta_2, \ldots) \) then the series is assumed to be

\[ x_t = \mu + 1_{\text{root}} x_{t-1} + \theta_1 e_t + \theta_2 e_{t-1} + \ldots \]

where \( 1_{\text{root}} = 1 \) if the option `root=1` and \( 1_{\text{root}} = 0 \) if the option `root=0`, and \( e_t \) is a white noise.

If `drift=TRUE` the drift adjusted series is obtained as

\[ \tilde{x}_t = x_t - t \left( \frac{x_T - x_1}{T - 1} \right), \quad t = 0, 1, \ldots, T - 1 \]

where \( \tilde{x}_t \) is the undrifting series.

**Value**

A "mFilter" object (see `mFilter`).

**Author(s)**

Mehmet Balcilar, <mbalcilar@yahoo.com>
References


See Also

mFilter, bwfilter, bkfilter, hpfilter, trfilter

Examples

```r
## library(mFilter)

data(unemp)

opar <- par(no.readonly=TRUE)

unemp.cf <- cffilter(unemp)
plot(unemp.cf)

unemp.cf1 <- cffilter(unemp, drift=TRUE, root=TRUE)
unemp.cf2 <- cffilter(unemp, pl=8, pu=40, drift=TRUE, root=TRUE)
unemp.cf3 <- cffilter(unemp, pl=2, pu=60, drift=TRUE, root=TRUE)
unemp.cf4 <- cffilter(unemp, pl=2, pu=40, drift=TRUE, root=TRUE, theta=c(1,.4))

par(mfrow=c(2,1),mar=c(3,3,2,1),cex=.8)
plot(unemp.cf1$x,
main="Christiano-Fitzgerald filter of unemployment: Trend \n root=TRUE,drift=TRUE",
col=1, ylab="")
lines(unemp.cf1$trend,col=2)
lines(unemp.cf2$trend,col=3)
lines(unemp.cf3$trend,col=4)
lines(unemp.cf4$trend,col=5)
legend("topleft",legend=c("series", "pl=2, pu=32", "pl=8, pu=40", "pl=2, pu=60", 
"pl=2, pu=40, theta=.1,.4"), col=1:5, lty=rep(1,5), ncol=1)

plot(unemp.cf1$y,
main="Christiano-Fitzgerald filter of unemployment: Cycle \n root=TRUE,drift=TRUE",
col=2, ylab="", ylim=range(unemp.cf1$y))
lines(unemp.cf2$y,col=3)
lines(unemp.cf3$y,col=4)
lines(unemp.cf4$y,col=5)
```
Description

This function implements the Hodrick-Prescott for estimating cyclical and trend component of a time series. The function computes cyclical and trend components of the time series using a frequency cut-off or smoothness parameter.

Usage

\[ \text{hpfilter}(x, \text{freq=NULL}, \text{type=c("lambda","frequency"), drift=FALSE}) \]

Arguments

- **x**: a regular time series.
- **type**: character, indicating the filter type, "lambda", for the filter that uses smoothness penalty parameter of the Hodrick-Prescott filter (default), "frequency", for the filter that uses a frequency cut-off type Hodrick-Prescott filter. These are related by \( \lambda = (2 * \sin(\pi/frequency))^{-4} \).
- **freq**: integer, if type="lambda" then freq is the smoothing parameter (lambda) of the Hodrick-Prescott filter, if type="frequency" then freq is the cut-off frequency of the Hodrick-Prescott filter.
- **drift**: logical, FALSE if no drift in time series (default), TRUE if drift in time series.

Details

Almost all filters in this package can be put into the following framework. Given a time series \( \{x_t\}_{t=1}^T \) we are interested in isolating component of \( x_t \), denoted \( y_t \) with period of oscillations between \( p_l \) and \( p_u \), where \( 2 \leq p_l < p_u < \infty \).

Consider the following decomposition of the time series

\[ x_t = y_t + \bar{x}_t \]

The component \( y_t \) is assumed to have power only in the frequencies in the interval \( \{(a, b) \cup (-a, -b)\} \in (-\pi, \pi) \). \( a \) and \( b \) are related to \( p_l \) and \( p_u \) by

\[ a = \frac{2\pi}{p_u}, \quad b = \frac{2\pi}{p_l} \]

If infinite amount of data is available, then we can use the ideal bandpass filter

\[ y_t = B(L)x_t \]
where the filter, \( B(L) \), is given in terms of the lag operator \( L \) and defined as

\[
B(L) = \sum_{j=-\infty}^{\infty} B_j L^j, \quad L^k x_t = x_{t-k}
\]

The ideal bandpass filter weights are given by

\[
B_j = \frac{\sin(jb) - \sin(ja)}{\pi j}, \quad B_0 = \frac{b - a}{\pi}
\]

The Hodrick-Prescott filter obtains the filter weights \( \hat{B}_j \) as a solution to

\[
\hat{B}_j = \arg \min \{ \sum_{t=1}^{T} (y_t - \hat{y}_t)^2 + \lambda \sum_{t=2}^{T-1} (\hat{y}_{t+1} - 2\hat{y}_t + \hat{y}_{t-1})^2 \}
\]

The Hodrick-Prescott filter is a finite data approximation with following moving average weights

\[
\hat{B}_j = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{4\lambda(1 - \cos(\omega))}{1 + 4\lambda(1 - \cos(\omega))^2} e^{i\omega j} d\omega
\]

If \( \text{drift} = \text{TRUE} \) the drift adjusted series is obtained as

\[
\tilde{x}_t = x_t - t \left( \frac{x_T - x_1}{T - 1} \right), \quad t = 0, 1, \ldots, T - 1
\]

where \( \tilde{x}_t \) is the undrifted series.

Value

A "mFilter" object (see \texttt{mFilter}).

Author(s)

Mehmet Balciar, \texttt{mbalcilar@yahoo.com}

References


**See Also**

mFilter, bwfilter, cffilter, bkfilter, trfilter

**Examples**

```r
## library(mFilter)

data(unemp)

opar <- par(no.readonly=TRUE)

unemp.hp <- hpfilter(unemp)
plot(unemp.hp)

unemp.hp1 <- hpfilter(unemp, drift=TRUE)
unemp.hp2 <- hpfilter(unemp, freq=800, drift=TRUE)
unemp.hp3 <- hpfilter(unemp, freq=12, type="frequency", drift=TRUE)
unemp.hp4 <- hpfilter(unemp, freq=52, type="frequency", drift=TRUE)

par(mfrow=c(2,1),mar=c(3,3,2,1),cex=.8)
plot(unemp.hp$trend,col=2)
lines(unemp.hp1$trend,col=3)
lines(unemp.hp2$trend,col=4)
lines(unemp.hp4$trend,col=5)
legend("topleft",legend=c("series", "lambda=1600", "lambda=800", 
"freq=12", "freq=52"), col=1:5, lty=rep(1,5), ncol=1)

plot(unemp.hp$cycle, main="Hodrick-Prescott filter of unemployment: Cycle,drift=TRUE", 
col=2, ylab="", ylim=range(unemp.hp4$cycle,na.rm=TRUE))
lines(unemp.hp2$cycle,col=3)
lines(unemp.hp3$cycle,col=4)
lines(unemp.hp4$cycle,col=5)
## legend("topleft",legend=c("lambda=1600", "lambda=800", 
## "freq=12", "freq=52"), col=1:5, lty=rep(1,5), ncol=1)

par(opar)
```

---

**mFilter**

*Decomposition of a time series into trend and cyclical components using various filters*

**Description**

mFilter is a generic function for filtering time series data. The function invokes particular filters which depend on filter type specified via its argument filter. The filters implemented in the package mFilter package are useful for smoothing, and estimating tend and cyclical components. Some
of these filters are commonly used in economics and finance for estimating cyclical component of
time series.

The `mFilter` currently applies only to time series objects. However a default method is available
and should work for any `numeric` or `vector` object.

**Usage**

```r
mFilter(x, ...)

## Default S3 method:
mFilter(x, ...)

## S3 method for class 'ts'
mFilter(x, filter=c("HP","BK","CF","BW","TR"), ...)
```

**Arguments**

- `x`: a regular a time series.
- `filter`: filter type, the filter types are "HP" (Hodrick-Prescott), "BK" (Baxter-King),
  "CF" (Christiano-Fitzgerald), "BW" (Butterworth), and "TR" (trigonometric re-
gression).
- `...`: Additional arguments to pass to the relevant filter functions. These are passed
to `hpfilter`, `bkfilter`, `cffilter`, `bwfilter`, and `trfilter`, respectively for
the "HP", "BK", "CF", "BW", and "TR" filters.

**Details**

The default behaviour is to apply the default filter to `ts` objects.

**Value**

An object of class "mFilter".

The function `summary` is used to obtain and print a summary of the results, while the function `plot`
produces a plot of the original series, the trend, and the cyclical components. The function `print`
is also available for displaying estimation results.

The generic accessor functions `fitted` and `residuals` extract estimated trend and cyclclical com-
ponets of an "mFilter" object, respectively.

An object of class "mFilter" is a list containing at least the following elements:

- `cycle`: Estimated cyclical (irregular) component of the series.
- `trend`: Estimated trend (smooth) component of the series.
- `fmatrix`: The filter matrix applied to original series.
- `method`: The method, if available, for the filter type applied.
- `type`: The filter type applied to the series.
- `call`: Call to the function.
- `title`: The title for displaying results.
- `xname`: Name of the series passed to `mFilter` for filtering.
x  The original or drift adjusted, if drift=TRUE, time series passed to the mFilter.

Following additional elements may exists depending on the type of filter applied:

  nfix  Length or order of the fixed length filters.
  pl   Minimum period of oscillation of desired component (2<=pl).
  pu   Maximum period of oscillation of desired component (2<=pl<pu<infinity).
  lambda  Lambda (smoothness) parameter of the HP filter.
  root  Whether time series has a unit root, TRUE or FALSE (default).
  drift  Whether time series has drift, TRUE or FALSE (default).
  theta  MA coefficients for time series model, used in "CF" filter.

Author(s)

  Mehmet Balcilar, mbalcilar@yahoo.com

See Also

  Other functions which return objects of class "mFilter" are bkfilter, bwfilter,cffilter, bkfilter, trfilter. Following functions apply the relevant methods to an object of the "mFilter" class: print.mFilter, summary.mFilter, plot.mFilter, fitted.mFilter, residuals.mFilter.

Examples

```r
## library(mFilter)

data(unemp)

opar <- par(no.readonly=TRUE)

unemp.hp <- mFilter(unemp,filter="HP")  # Hodrick-Prescott filter
print(unemp.hp)
summary(unemp.hp)
residuals(unemp.hp)
fitted(unemp.hp)
plot(unemp.hp)

unemp.bk <- mFilter(unemp,filter="BK")  # Baxter-King filter
unemp.cf <- mFilter(unemp,filter="CF")  # Christiano-Fitzgerald filter
unemp.bw <- mFilter(unemp,filter="BW")  # Butterworth filter
unemp.tr <- mFilter(unemp,filter="TR")  # Trigonometric regression filter

par(mfrow=c(2,1),mar=c(3,3,2,1),cex=.8)
plot(unemp,main="Unemployment Series & Estimated Trend", col=1, ylab="")
lines(unemp.hp$trend,col=2)
lines(unemp.bk$trend,col=3)
lines(unemp.cf$trend,col=4)
lines(unemp.bw$trend,col=5)
lines(unemp.tr$trend,col=6)
```
legend("topleft", legend=c("series", "HP", "BK", "CF", "BW", "TR"),
    col=1:6, lty=rep(1, 6), ncol=2)

plot(unemp.hp$cycle, main="Estimated Cyclical Component",
    ylim=c(-2, 2.5), col=2, ylab="")
lines(unemp.bk$cycle, col=3)
lines(unemp.cf$cycle, col=4)
lines(unemp.bw$cycle, col=5)
lines(unemp.tr$cycle, col=6)
## legend("topleft", legend=c("HP", "BK", "CF", "BW", "TR"),
## col=2:6, lty=rep(1, 5), ncol=2)

unemp.cf1 <- mFilter(unemp, filter="CF", drift=TRUE, root=TRUE)
unemp.cf2 <- mFilter(unemp, filter="CF", pl=8, pu=40, drift=TRUE, root=TRUE)
unemp.cf3 <- mFilter(unemp, filter="CF", pl=2, pu=60, drift=TRUE, root=TRUE)
unemp.cf4 <- mFilter(unemp, filter="CF", pl=2, pu=40, theta=c(.1, .4))

plot(unemp,
    main="Christiano-Fitzgerald filter of unemployment: Trend 
         root=TRUE, drift=TRUE",
    col=1, ylab="")
lines(unemp.cf1$trend, col=2)
lines(unemp.cf2$trend, col=3)
lines(unemp.cf3$trend, col=4)
lines(unemp.cf4$trend, col=5)
legend("topleft", legend=c("series", "pl=2, pu=32", "pl=8, pu=40",
                          "pl=2, pu=60", "pl=2, pu=40, theta=1.4"),
    col=1:5, lty=rep(1, 5), ncol=1)

plot(unemp.cf1$cycle,
    main="Christiano-Fitzgerald filter of unemployment: Cycle 
         root=TRUE, drift=TRUE",
    col=2, ylab="", ylim=range(unemp.cf3$cycle))
lines(unemp.cf2$cycle, col=3)
lines(unemp.cf3$cycle, col=4)
lines(unemp.cf4$cycle, col=5)
## legend("topleft", legend=c("pl=2, pu=32", "pl=8, pu=40", "pl=2, pu=60",
## "pl=2, pu=40, theta=1.4"), col=2:5, lty=rep(1, 4), ncol=2)

par(opar)

mFilter-methods

Methods for mFilter objects

Description

Common methods for all mFilter objects usually created by the mFilter function.

Usage

## S3 method for class 'mFilter'
residuals(object, ...)

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

## S3 method for class 'mFilter'
print(x, digits = max(3,getOption("digits") - 3), ...)

## S3 method for class 'mFilter'
plot(x, reference.grid = TRUE, col = "steelblue", ask=interactive(), ...)

## S3 method for class 'mFilter'
summary(object, digits = max(3,getOption("digits") - 3), ...)

### Arguments

- **object, x**: an object of class "mFilter"; usually, a result of a call to `mFilter`.
- **digits**: number of digits used for printing (see `print`).
- **col**: color of the graph (see `plot`).
- **ask**: logical. if TRUE the user is asked for input before a new graph drawn in an interactive session (see `interactive`).
- **reference.grid**: logical. if true grid lines are drawn.
- **...**: further arguments passed to or from other methods.

### Value

for residuals and fitted a univariate time series; for `plot`, `print`, and `summary` the "mFilter" object.

### Author(s)

Mehmet Balcilar, <mbalcilar@yahoo.com>

### See Also

- `mFilter` for the function that returns an objects of class "mFilter". Other functions which return objects of class "mFilter" are `bkfilter`, `bwfilter`, `cffilter`, `bkfilter`, `trfilter`.

### Examples

```r
## library(mFilter)

data(unemp)

opar <- par(no.readonly=TRUE)

unemp.hp <- mFilter(unemp,filter="HP")  # Hodrick-Prescott filter
print(unemp.hp)
summary(unemp.hp)
residuals(unemp.hp)
fitted(unemp.hp)
plot(unemp.hp)

par(opar)
```
Description

This function uses trigonometric regression filter for estimating cyclical and trend components of a time series. The function computes cyclical and trend components of the time series using a lower and upper cut-off frequency in the spirit of a band pass filter.

Usage

```
trfilter(x, pl=NULL, pu=NULL, drift=FALSE)
```

Arguments

- `x`: a regular time series.
- `pl`: integer. minimum period of oscillation of desired component (pl<=2).
- `pu`: integer. maximum period of oscillation of desired component (2<=pl<pu<infinity).
- `drift`: logical, FALSE if no drift in time series (default), TRUE if drift in time series.

Details

Almost all filters in this package can be put into the following framework. Given a time series \( \{x_t\}_{t=1}^T \) we are interested in isolating component of \( x_t \), denoted \( y_t \) with period of oscillations between \( p_l \) and \( p_u \), where \( 2 \leq p_l < p_u < \infty \).

Consider the following decomposition of the time series

\[
x_t = y_t + \bar{x}_t
\]

The component \( y_t \) is assumed to have power only in the frequencies in the interval \( \{(a, b) \cup (-a, -b)\} \in (-\pi, \pi) \). \( a \) and \( b \) are related to \( p_l \) and \( p_u \) by

\[
a = \frac{2\pi}{p_u}, \quad b = \frac{2\pi}{p_l}
\]

If infinite amount of data is available, then we can use the ideal bandpass filter

\[
y_t = B(L)x_t
\]

where the filter, \( B(L) \), is given in terms of the lag operator \( L \) and defined as

\[
B(L) = \sum_{j=-\infty}^{\infty} B_j L^j, \quad L^k x_t = x_{t-k}
\]

The ideal bandpass filter weights are given by

\[
B_j = \frac{\sin(jb) - \sin(ja)}{\pi j}
\]
Let $T$ be even and define $n_1 = T/p_u$ and $n_2 = T/p_l$. The trigonometric regression filter is based on the following relation

$$y_t = \sum_{j=n_2}^{n_1} \{a_j \cos(\omega_j t) + b_j \sin(\omega_j t)\}$$

where $a_j$ and $b_j$ are the coefficients obtained by regressing $x_t$ on the indicated sine and cosine functions. Specifically,

$$a_j = \frac{T}{\pi} \sum_{t=1}^{T} \cos(\omega_j t)x_t, \quad \text{for } j = 1, \ldots, T/2 - 1$$
$$a_j = \frac{T}{\pi} \sum_{t=1}^{T} \cos(\pi t)x_t, \quad \text{for } j = T/2$$

and

$$b_j = \frac{T}{\pi} \sum_{t=1}^{T} \sin(\omega_j t)x_t, \quad \text{for } j = 1, \ldots, T/2 - 1$$
$$b_j = \frac{T}{\pi} \sum_{t=1}^{T} \sin(\pi t)x_t, \quad \text{for } j = T/2$$

Let $\hat{B}(L)x_t$ be the trigonometric regression filter. It can be showed that $\hat{B}(1) = 0$, so that $\hat{B}(L)$ has a unit root for $t = 1, 2, \ldots, T$. Also, when $\hat{B}(L)$ is symmetric, it has a second unit root in the middle of the data for $t$. Therefore it is important to drift adjust data before it is filtered with a trigonometric regression filter.

If drift=TRUE the drift adjusted series is obtained as

$$\tilde{x}_t = x_t - t \left(\frac{x_T - x_1}{T - 1}\right), \quad t = 0, 1, \ldots, T - 1$$

where $\tilde{x}_t$ is the undrifted series.

Value

A "mFilter" object (see mFilter).

Author(s)

Mehmet Balcilar, <mbalcilar@yahoo.com>

References


See Also

mFilter, hpfilter, cffilter, bkfilter, bwfilter

Examples

```r
## library(mFilter)

data(unemp)

opar <- par(no.readonly=TRUE)

unemp.tr <- trfilter(unemp, drift=TRUE)
plot(unemp.tr)
unemp.tr1 <- trfilter(unemp, drift=TRUE)
unemp.tr2 <- trfilter(unemp, pl=8, pu=40, drift=TRUE)
unemp.tr3 <- trfilter(unemp, pl=2, pu=60, drift=TRUE)
unemp.tr4 <- trfilter(unemp, pl=2, pu=40, drift=TRUE)

par(mfrow=c(2,1),mar=c(3,3,2,1),cex=.8)
plot(unemp.tr1$x,
main="Trigonometric regression filter of unemployment: Trend, drift=TRUE",
col=1, ylab="")
lines(unemp.tr1$trend,col=2)
lines(unemp.tr2$trend,col=3)
lines(unemp.tr3$trend,col=4)
lines(unemp.tr4$trend,col=5)
legend("topleft",legend=c("series", "pl=2, pu=32", "pl=8, pu=40",
"pl=2, pu=60", "pl=2, pu=40"), col=1:5, lty=rep(1,5), ncol=1)

plot(unemp.tr1$cycle,
main="Trigonometric regression filter of unemployment: Cycle,drift=TRUE",
col=2, ylab="", ylim=range(unemp.tr3$cycle,na.rm=TRUE))
lines(unemp.tr2$cycle,col=3)
lines(unemp.tr3$cycle,col=4)
lines(unemp.tr4$cycle,col=5)
## legend("topleft",legend=c("pl=2, pu=32", "pl=8, pu=40", "pl=2, pu=60",
## "pl=2, pu=40"), col=1:5, lty=rep(1,5), ncol=1)

par(opar)
```

---

**unemp**  

*US Quarterly Unemployment Series*

Description


*number of observations*: 168

*observation*: country

*country*: United States
Usage

data(unemp)

Format

A time series containing:

- **unemp** unemployment rate (average of months in quarter)

Author(s)

Mehmet Balcilar, <mbalcilar@yahoo.com>

Source


References


Examples

```r
## library(mFilter)

data(unemp)

unemp.hp <- mFilter(unemp,filter="HP")  # Hodrick-Prescott filter
unemp.bk <- mFilter(unemp,filter="BK")  # Baxter-King filter
unemp.cf <- mFilter(unemp,filter="CF")  # Christiano-Fitzgerald filter

opar <- par(no.readonly=TRUE)
par(mfrow=c(2,1), mar=c(3,3,2,1))
plot(unemp,main="Unemployment Series & Estimated Trend",col=1,ylab="")
lines(unemp.hp$trend,col=2)
lines(unemp.bk$trend,col=3)
lines(unemp.cf$trend,col=4)
legend("topleft",legend=c("series", "HP", "BK", "CF"),col=1:4,
  lty=rep(1,4),ncol=2)

plot(unemp.hp$cycle,main="Estimated Cyclical Component",col=2,
  ylim=c(-2,2),ylab="")
lines(unemp.bk$cycle,col=3)
lines(unemp.cf$cycle,col=4)
legend("topleft",legend=c("HP", "BK", "CF"),col=2:4,lty=rep(1,3),ncol=2)
par(opar)
```
Index

* Topic **datasets**
  - unemp, 23

* Topic **loess**
  - bkfilter, 5
  - bwfilter, 8
  - cffilter, 11
  - hpfilter, 14
  - mFilter, 16
  - mFilter-methods, 19
  - mFilter-package, 2
  - trfilter, 21

* Topic **nonparametric**
  - bkfilter, 5
  - bwfilter, 8
  - cffilter, 11
  - hpfilter, 14
  - mFilter, 16
  - mFilter-methods, 19
  - mFilter-package, 2
  - trfilter, 21

* Topic **smooth**
  - bkfilter, 5
  - bwfilter, 8
  - cffilter, 11
  - hpfilter, 14
  - mFilter, 16
  - mFilter-methods, 19
  - mFilter-package, 2
  - trfilter, 21

* Topic **ts**
  - bkfilter, 5
  - bwfilter, 8
  - cffilter, 11
  - hpfilter, 14
  - mFilter, 16
  - mFilter-methods, 19
  - mFilter-package, 2
  - trfilter, 21

bkfilter, 5, 5, 10, 13, 16, 18, 20, 23
bwfilter, 5, 7, 8, 13, 16, 18, 20, 23
cffilter, 5, 7, 10, 11, 16, 18, 20, 23
fitted.mFilter, 18
fitted.mFilter (mFilter-methods), 19
hpfilter, 5, 7, 10, 13, 14, 23
interactive, 20
mFilter, 5–7, 9, 10, 12, 13, 15, 16, 16, 20, 22, 23
mFilter-methods, 19
mFilter-package, 2
numeric, 17
plot, 20
plot.mFilter, 18
plot.mFilter (mFilter-methods), 19
print, 20
print.mFilter, 18
print.mFilter (mFilter-methods), 19
residuals.mFilter, 18
residuals.mFilter (mFilter-methods), 19
summary.mFilter, 18
summary.mFilter (mFilter-methods), 19
trfilter, 5, 7, 10, 13, 16, 18, 20, 21
ts, 17
unemp, 23
vector, 17