Filters

2024-10-07

Introduction

To extract information from a time series or -equivalently - to remove unwanted noises, we often apply linear filters. The filters can be built following quite different approaches. Using an annual time series, we consider different solutions to extract its trend. Similar derivations are used in seasonal adjustment procedures. By considering only the trend extraction, we can focus on the most relevant aspects of the various approaches.

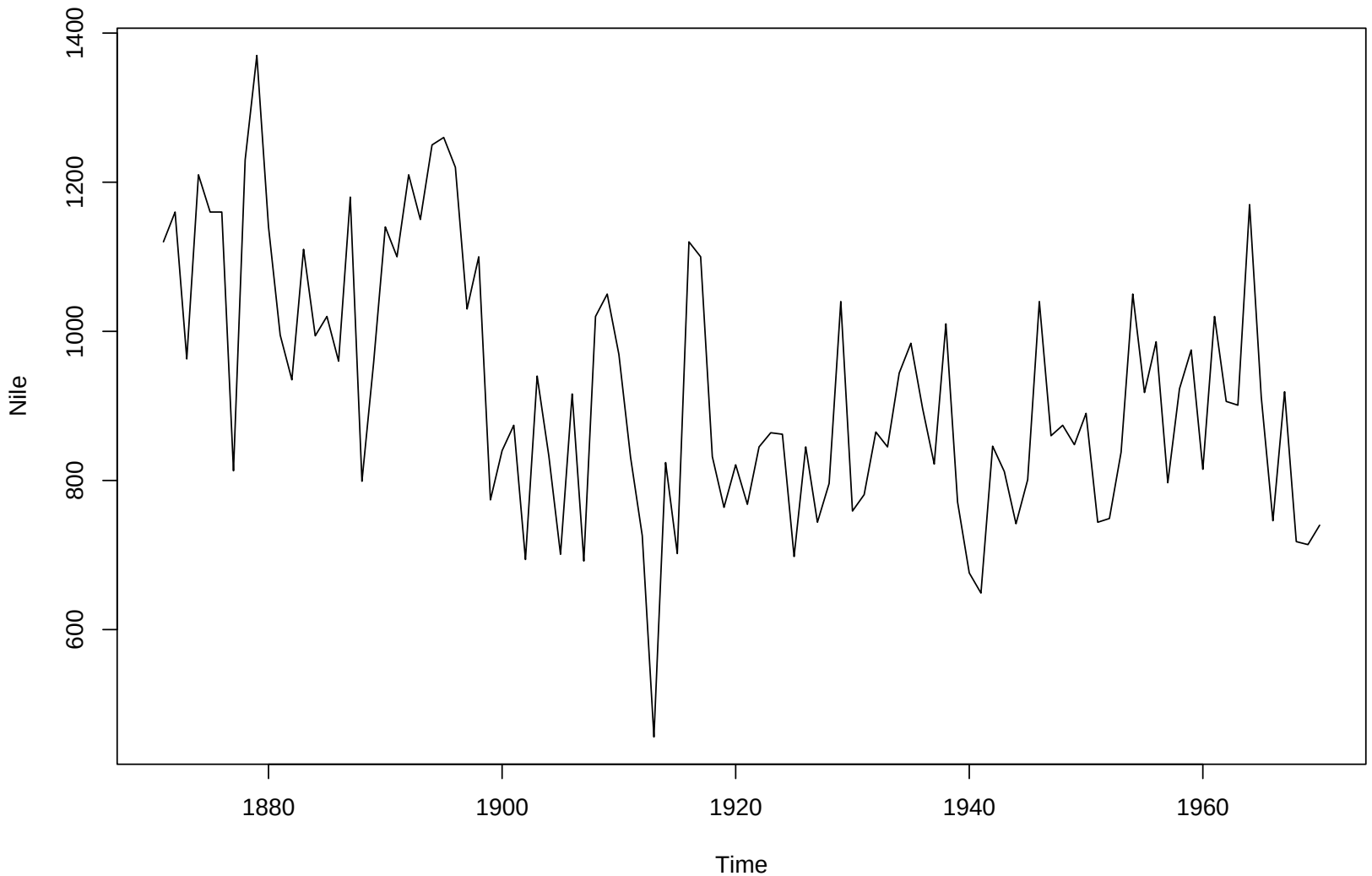
Outline

- Description of the Data
- Pre-specified filters
  - Henderson filters
- Local polynomials
- Mode-based filters
  - Structural models with fixed parameters (Hodrick-Prescott)
  - Parametric ARIMA and canonical decomposition

Data

Flow of the river Nule

```
plot(Nile)
```



Pre-specified filters

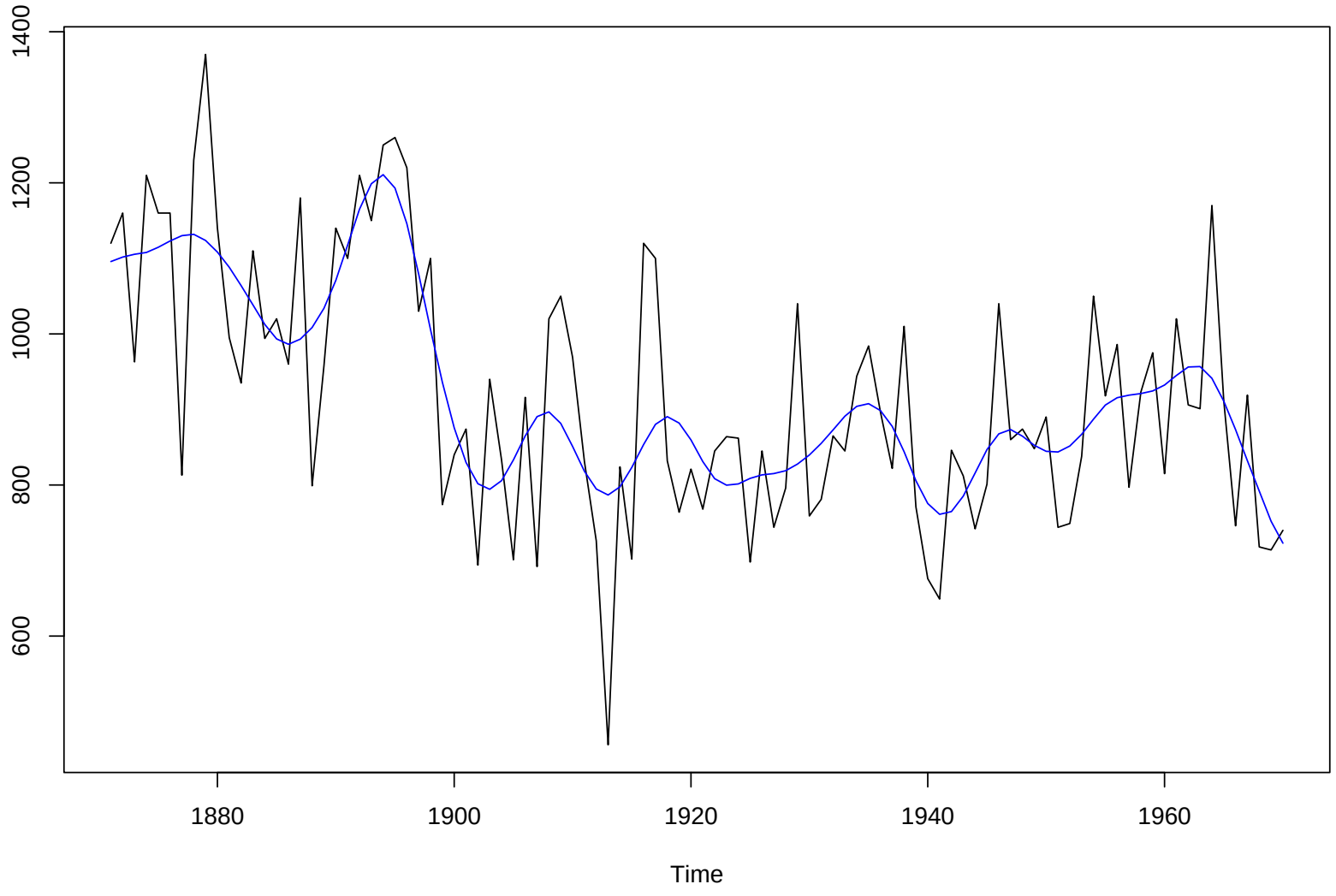
Filters based on well-defined mathematical properties

Example: Henderson filters

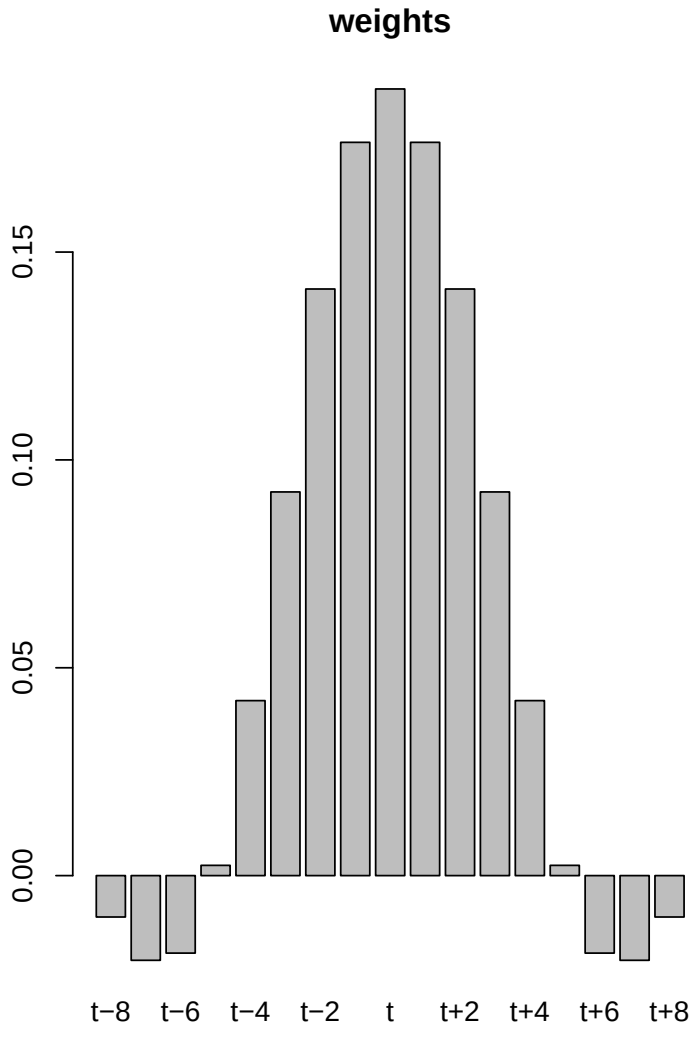
The Henderson filter $\{\omega_i\}_{-p \leq i \leq p}$ is defined by minimizing $\sum_{i=-p}^{p} (\Delta^3 \omega_i)^2$ under the constraints: $\sum_{i=-p}^{p} \omega_i = 1$, $\sum_{i=-p}^{p} i\omega_i = 0$ and $\sum_{i=-p}^{p} i^2 \omega_i = 0$

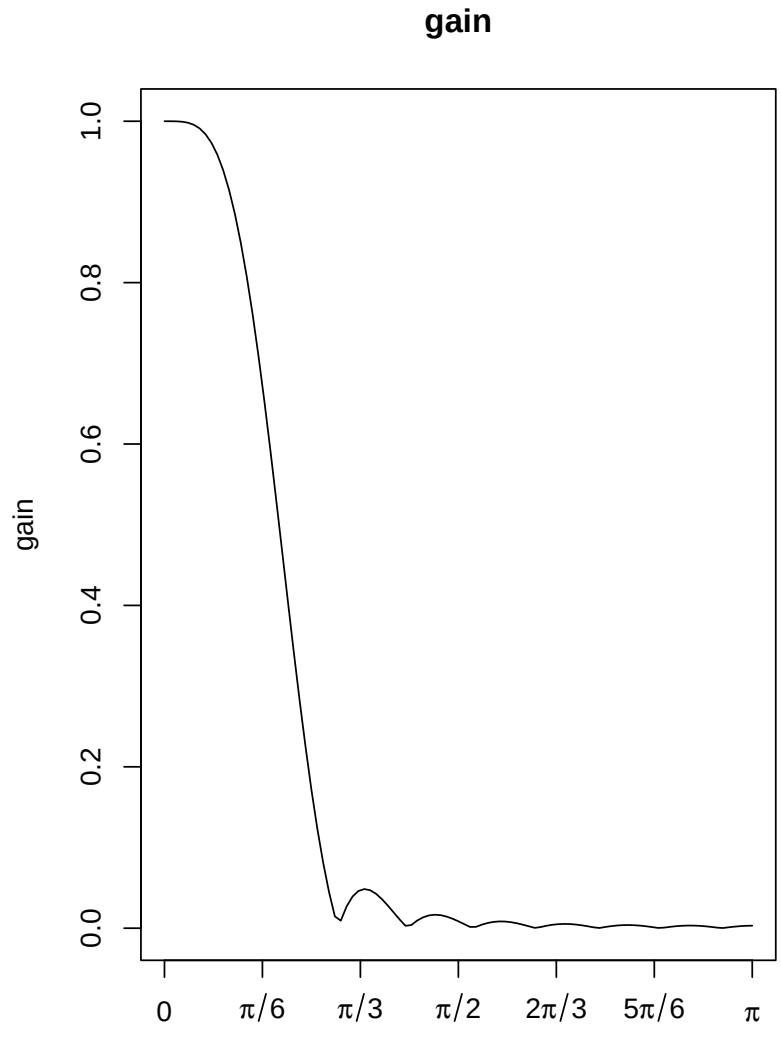
```
nh<-17
hnile<-rjd3x11plus::henderson(Nile, nh)
```

Applying Henderson filters extended by Musgrave filters (X11-like)



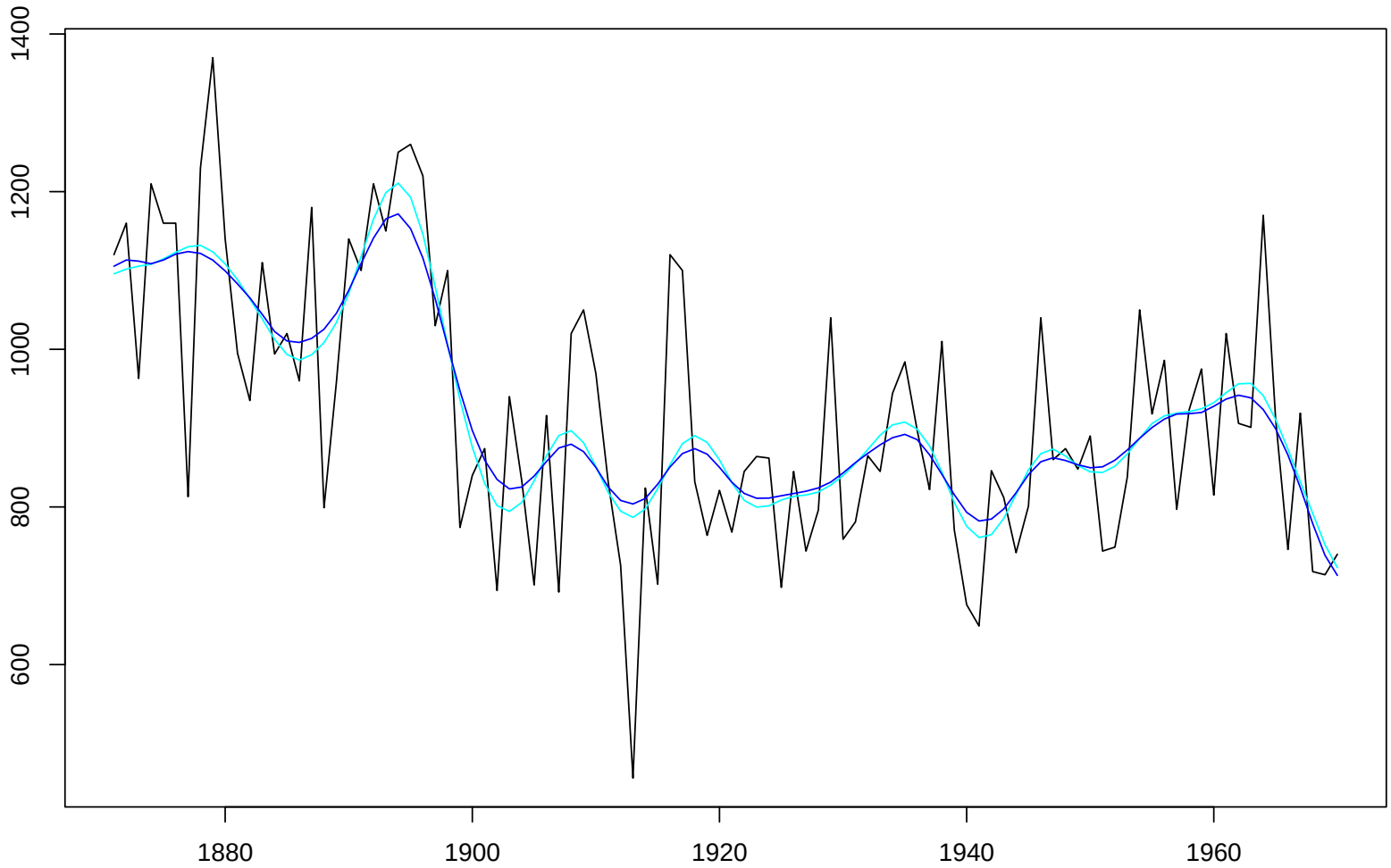
Henderson filter properties



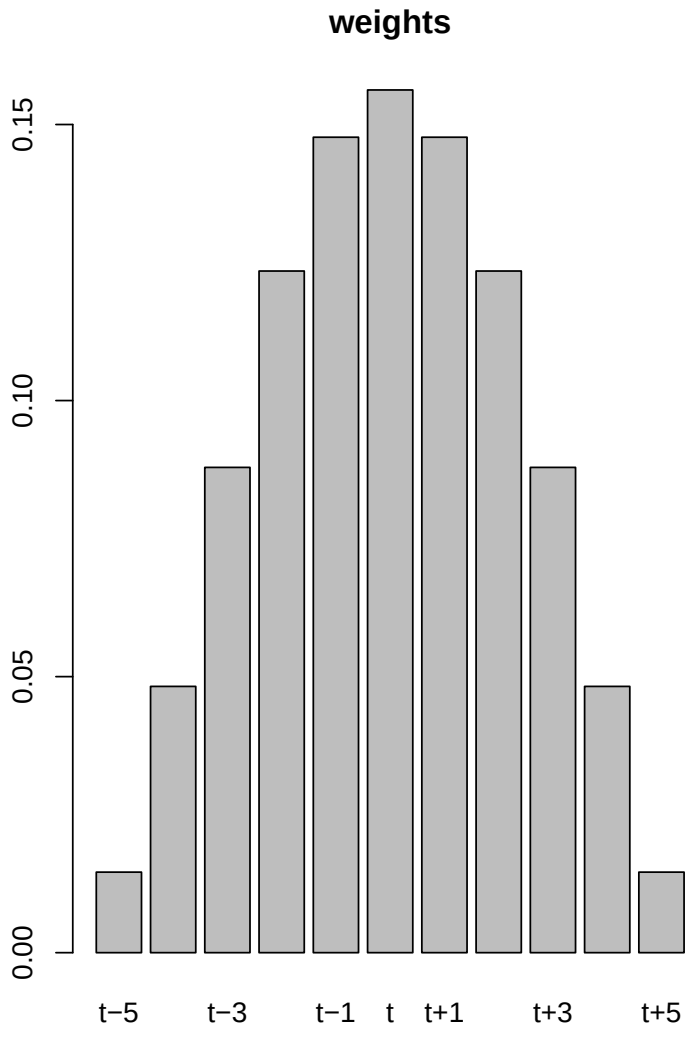


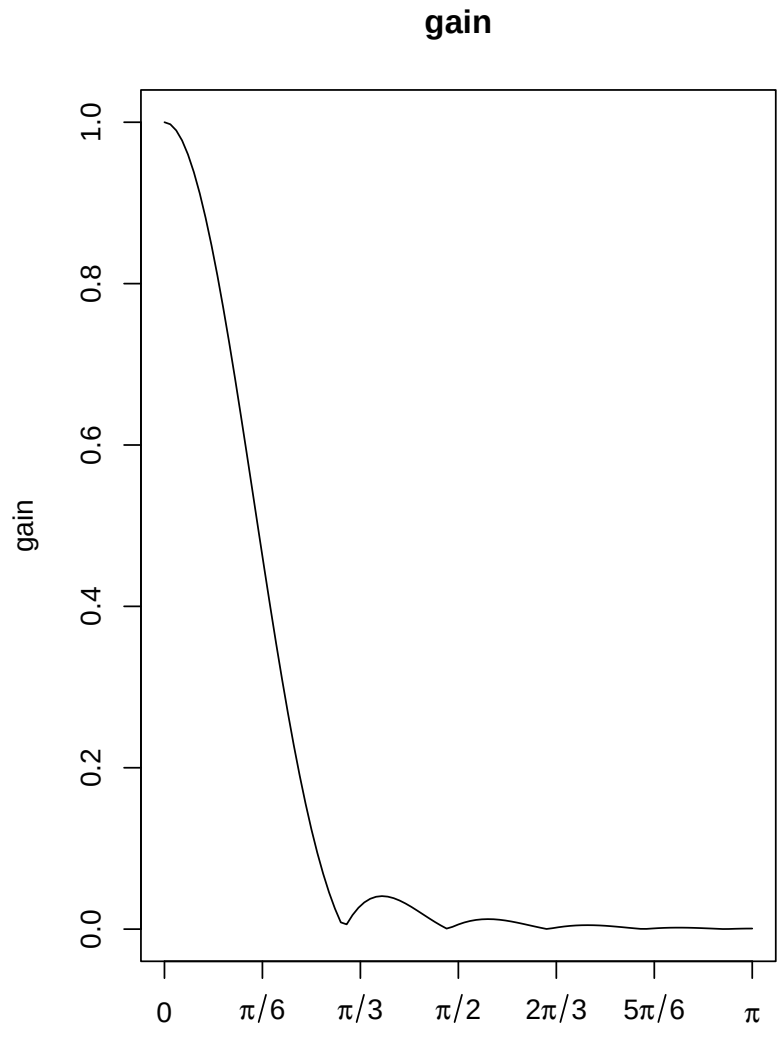
Local polynomials

For each time point, the smoothed value is defined by the regression with a linear trend on 11 periods, using a bi-weight kernel $\alpha(1 - x^2)^2)$



Loess filter properties





Model-based filters (I)

Structural model (fixed parameters)

$$
\begin{aligned}
y(t) &= T(t) + N(t) \\
\Delta^2 T(t) &= \epsilon_T(t) && \epsilon_T \sim N(0,1) \\
N(t) &= \epsilon_N(t) && \epsilon_N \sim N(0,\lambda)
\end{aligned}
$$

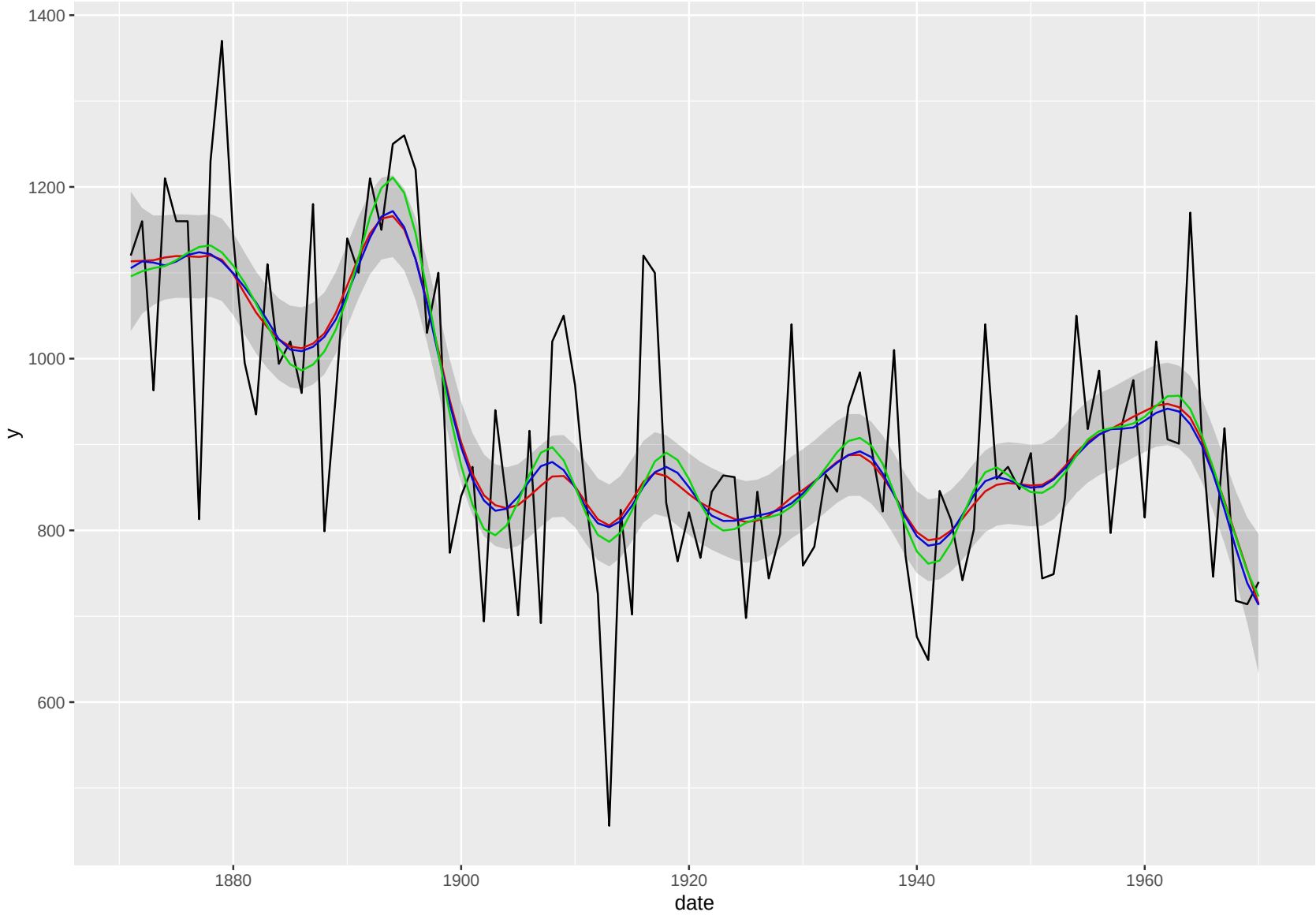
```
hp_ucm<-function(lambda=25){
  i2<-rjd3toolkit::arima_model("trend", delta = c(1,-2, 1))
  noise<-rjd3toolkit::arima_model(variance=lambda)
  ucm<-rjd3toolkit::ucarima_model(components=list(i2, noise))
}

ucm_estimate<-function(x, ucm, stdev=TRUE){
  jucm<-rjd3toolkit:::.r2jd_ucarima(ucm)
  jcmps<-rJava::.jcall("jdplus/toolkit/base/r/arima/UcarimaModels",
            "Ljdplus/toolkit/base/api/math/matrices/Matrix;", "estimate",
            as.numeric(x), jucm, as.logical(stdev))
  return(rjd3toolkit:::.jd2r_matrix(jcmps))
}

hp_estimate<-function(s, lambda=25){
  ucm<-hp_ucm(lambda = lambda)
  rslt<-ucm_estimate(s, ucm)
}
```

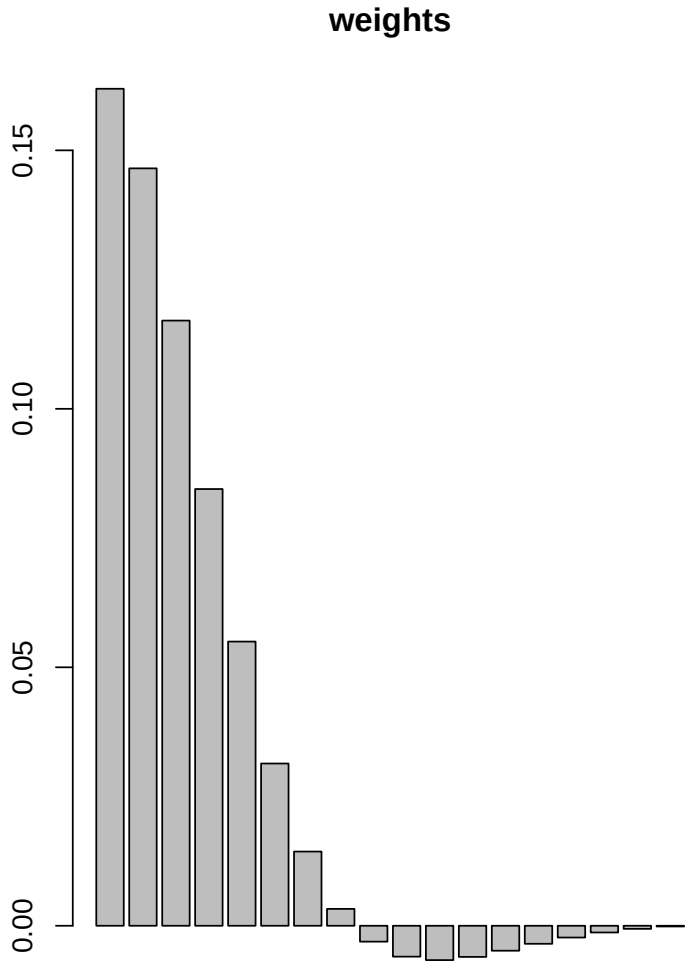
Model-based filters (I)

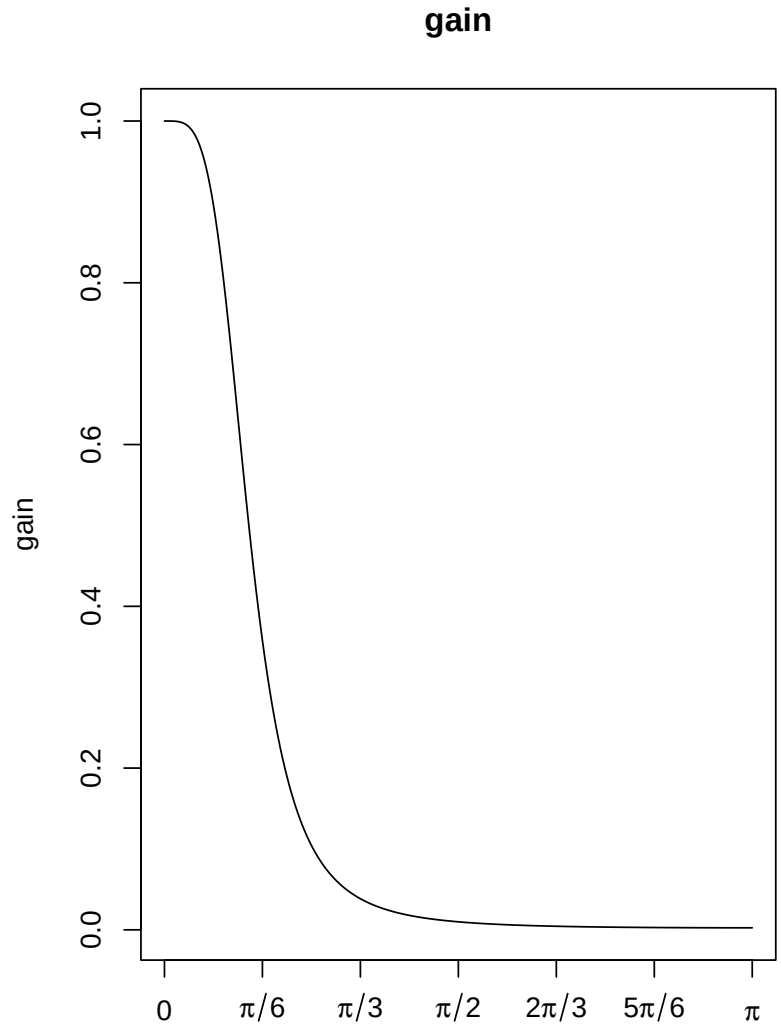
Structural model (fixed parameters)



Model-based filters properties

Structural model (fixed parameters)





Model-based filters properties

Canonical decomposition

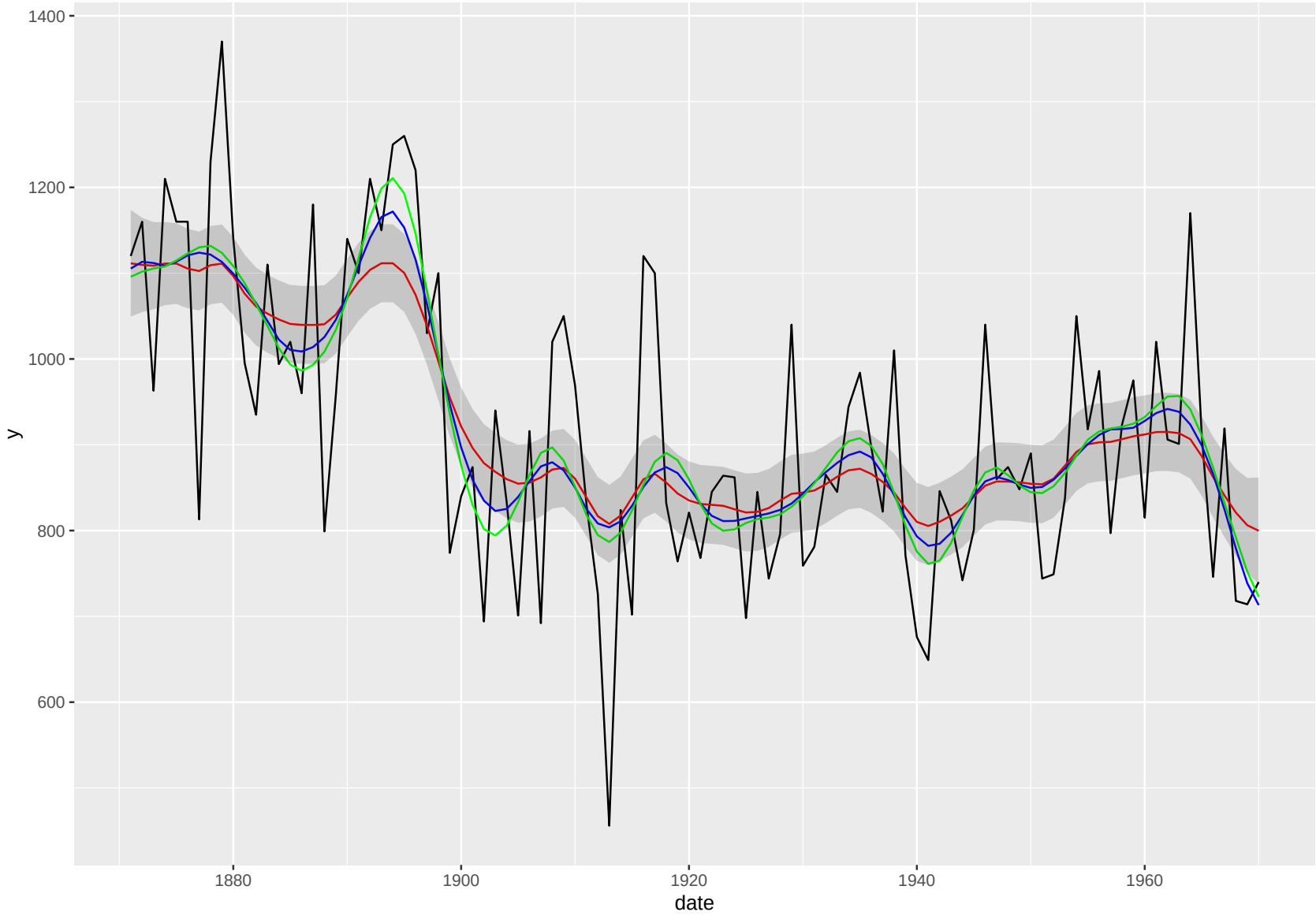
```
hp_ucm<-function(lambda=25){
  i2<-rjd3toolkit::arima_model("trend", delta = c(1,-2, 1))
  noise<-rjd3toolkit::arima_model(variance=lambda)
  ucm<-rjd3toolkit::ucarima_model(components=list(i2, noise))
}

ucm_estimate<-function(x, ucm, stdev=TRUE){
  jucm<-rjd3toolkit:::.r2jd_ucarima(ucm)
  jcmps<-rJava::.jcall("jdplus/toolkit/base/r/arima/UcarimaModels",
            "Ljdplus/toolkit/base/api/math/matrices/Matrix;", "estimate",
            as.numeric(x), jucm, as.logical(stdev))
  return(rjd3toolkit:::.jd2r_matrix(jcmps))
}

hp_estimate<-function(s, lambda=25){
  ucm<-hp_ucm(lambda = lambda)
  rslt<-ucm_estimate(s, ucm)
}
```

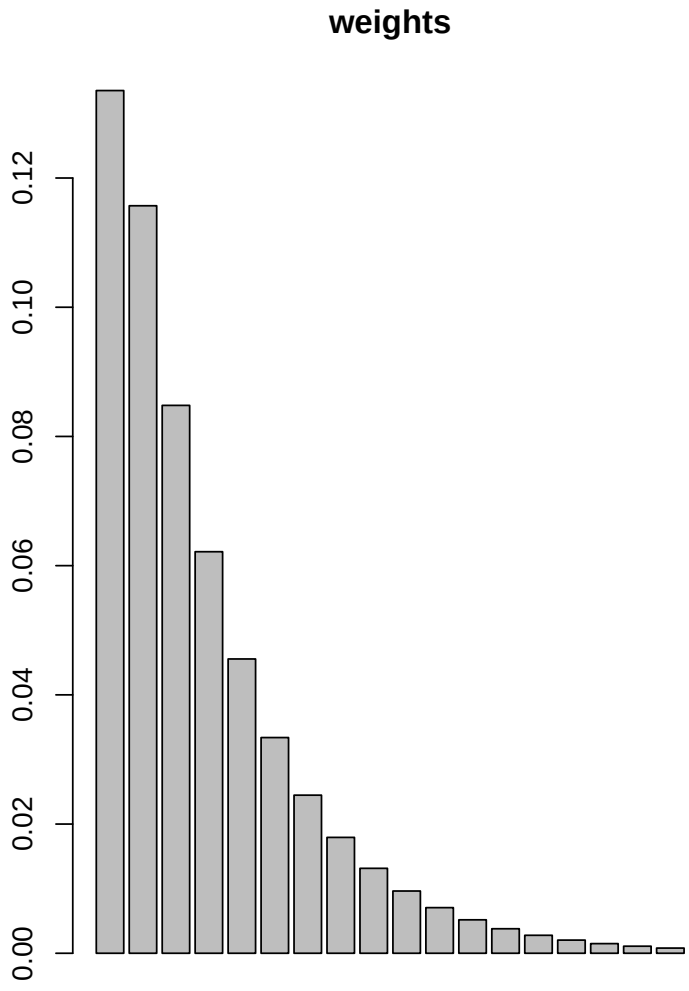
Model-based filters

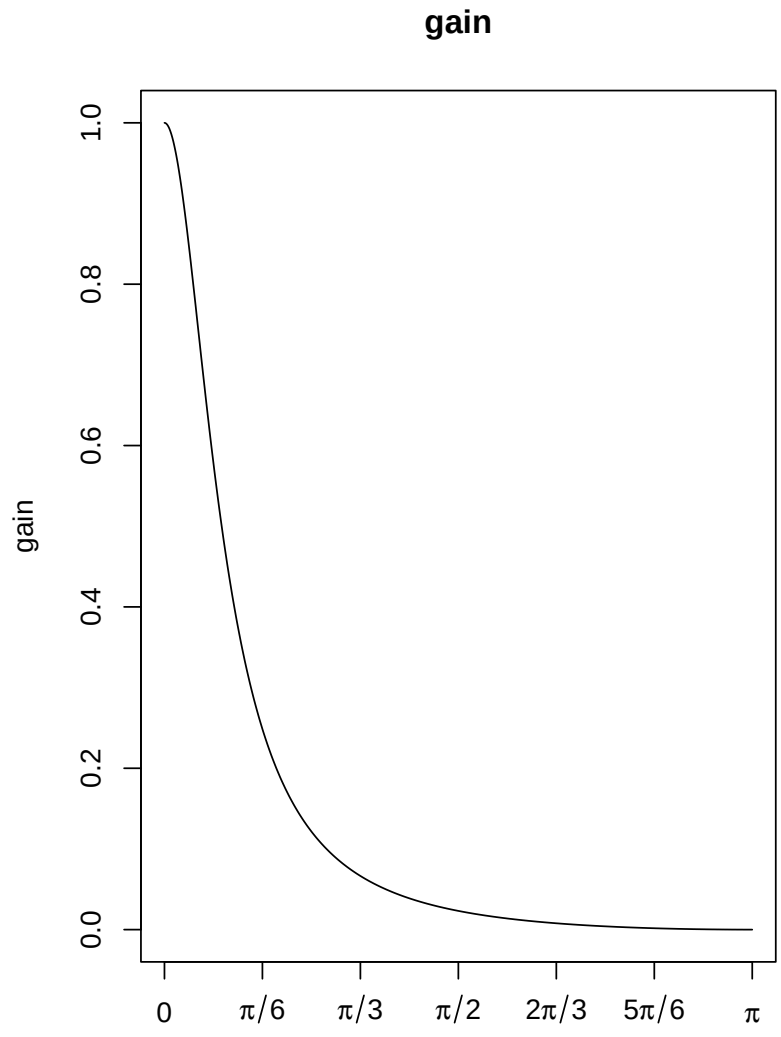
Canonical decomposition



Model-based filters properties

Canonical decomposition





Annexe: Differencing

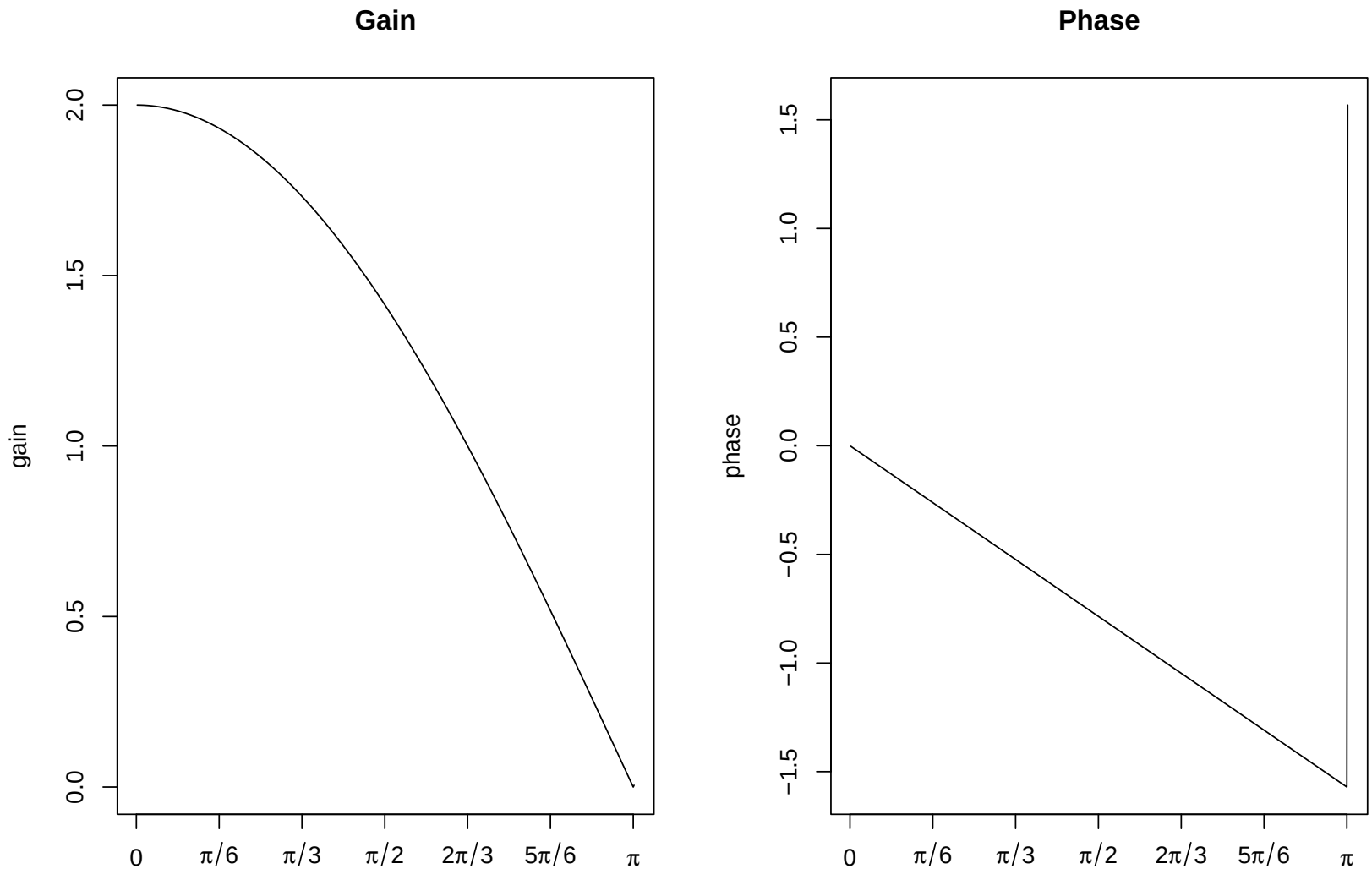
```
## Filter x(t)-x(t-1) = (1-B)x(t)

rf<-function(w){
  1+complex(real=cos(w), imaginary = -sin(w))
}

w<-seq(0:600)*(pi/600)
cw<-rf(w)

gain<-Mod(cw)
phase<-Arg(cw)
## or phase<-Arg(cw)/w
```

Properties


Gain
gain
2.0
1.5
1.0
0.5
0.0
0
π/6
π/3
π/2
2π/3
5π/6
π
Phase
phase
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0
π/6
π/3
π/2
2π/3
5π/6
π