Time Series: A First Course with Bootstrap Starter

Lesson 3-1: Nonparametric Smoothing

- The filtering of time series uses some concepts from nonparametric smoothing.

Nonparametric Regression

- We might think of the time series mean $\mu_t = \mathbb{E}[X_t]$ as an arbitary, smoothly varying function.
- A very simple case:
$$X_t = \mu_t + Z_t$$
with $Z_t \sim i.i.d.(0, \sigma^2)$.
- We may estimate μ_t via averaging over neighboring values, e.g.,
$$\widehat{\mu}_t = \frac{1}{2m+1} \sum_{k=-m}^{m} X_{t-k}.$$
This works because
$$\widehat{\mu}_t = \frac{1}{2m+1} \sum_{k=-m}^{m} \mu_{t-k} + \frac{1}{2m+1} \sum_{k=-m}^{m} Z_{t-k},$$
and the first term is approximately equal to μ_t if m is small and the mean is smooth (not changing too much over time). Also, the second term has mean zero and variance $\sigma^2/(2m+1)$, which tends to zero as m increases to ∞.
- So bias is lower for small m, but variance is lower for large m, creating a tension.

Kernel and Bandwidth

- Above, we have equal weights on the observations. We can have unequal weights, through the device of a *kernel*, which is a function that weights the data.
- The *bandwidth* m defines a neighborhood of values around time t, for which we use the kernel's weights.

- Large bandwidth yields more smoothing, with suppression of local features.
- Small bandwidth yields less smoothing, with higher variability.

Edge Effects

- We can only compute $\widehat{\mu}_t$ for $t = m+1, \ldots, n-m$. So the values for $t = 1, \ldots, m$ and $t = n-m+1, \ldots, n$ are "missing".
- Sometimes this so-called "edge effect" is addressed by designing asymmetric smoothing at the boundaries.
- Often the edge effects are ignored, when we are interested in the interior of the time series sample.

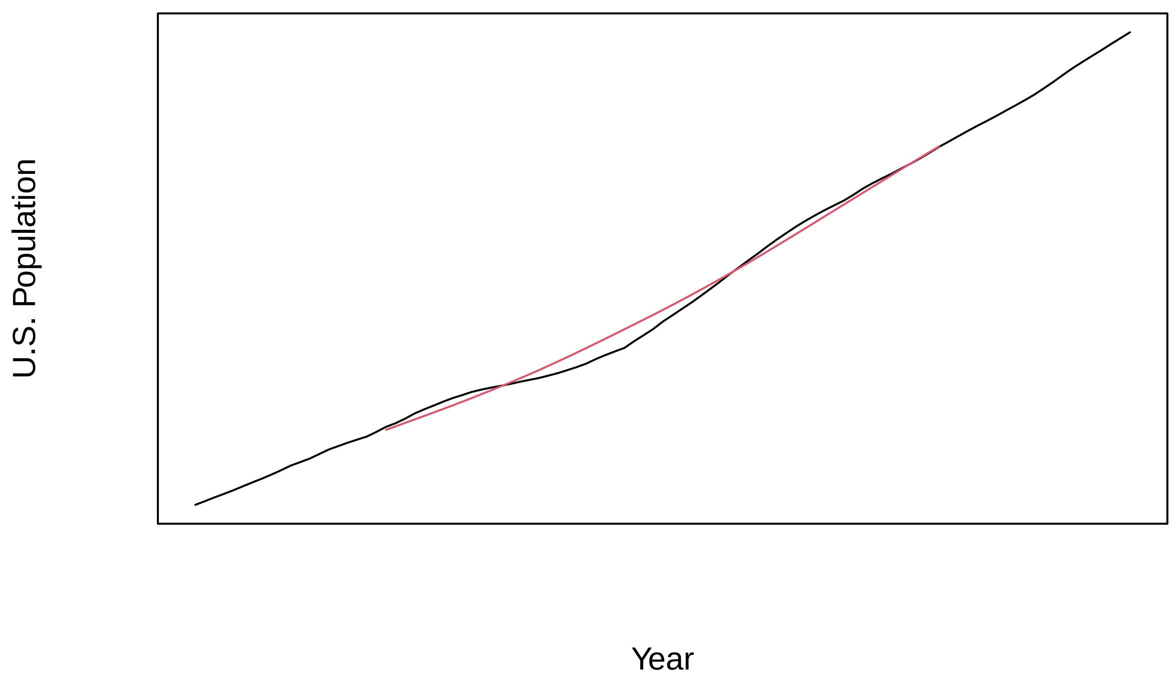
Nonparametric Regression for Population

- Apply the nonparametric regression technique with $m = 20$ to U.S. population data.
- First, we write a function to do local averaging.

```
simple.ma <- function(x,m)
{
    weights <- rep(1,2*m+1)/(2*m+1)
    n <- length(x)
    trend <- NULL
    for(t in (m+1):(n-m))
    {
        trend <- c(trend,sum(x[(t-m):(t+m)]*weights))
    }
    trend <- c(rep(NA,m),trend,rep(NA,m))
    return(trend)
}
```

- Then we apply this to the data.

```
pop <- read.table("USpop.dat")
pop <- ts(pop, start = 1901)
m <- 20
pop.trend <- simple.ma(pop,m)
plot(pop,xlab="Year",ylab="U.S. Population",col=1,lwd=1,yaxt="n",xaxt="n")
lines(ts(pop.trend,start=1901,frequency=1),lty=1,lwd=1,col=2)
```



Lesson 3-2: Linear Filters

- A *linear filter* maps an input time series $\{X_t\}$ to an output time series $\{Y_t\}$ by taking a linear combination of past, present, and future observations:

$$Y_t = \sum_{k \in \mathbb{Z}} \psi_k \, X_{t-k}.$$

The *filter coefficients* (or *filter weights*) are $\{\psi_k\}$. Note the convention that ψ_k weights an observation occurring k time points in the past (viewing time t as the present).

Example 3.2.5. Simple Moving Average

- Recall from nonparametric regression, we may estimate a slowly-changing mean via averaging over neighboring values:

$$\frac{1}{2m+1} \sum_{k=-m}^{m} X_{t-k}.$$

This weights the past m and the future m observations equally, and is an example of a linear filter. It also called a *moving average*, because the observations are averaged (over a window of $2m+1$ time points) in a way that moves over the time series. A *simple moving average* has equal weights (a general moving average could have unequal weights).

Linear Time Series

- A time series $\{Y_t\}$ is said to be *linear* if it is defined as the output of a linear filter (with coefficients $\{\psi_k\}$) applied to $\{X_t\}$, and $X_t \sim i.i.d.(\mu, \sigma^2)$.
- The mean of Y_t is

$$\mathbb{E}[Y_t] = \mu \sum_{k \in \mathbb{Z}} \psi_k.$$

- The autocovariance of $\{Y_t\}$ is

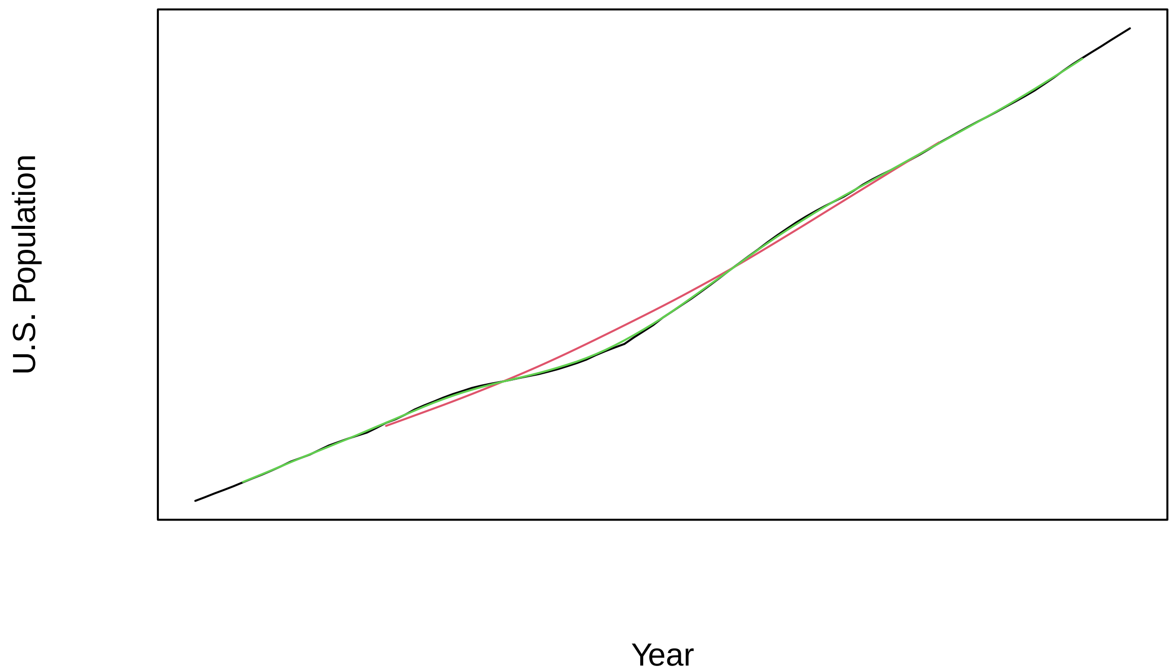
$$\mathrm{Cov}[Y_t, Y_{t-h}] = \sum_{j,k \in \mathbb{Z}} \psi_j \psi_k \mathrm{Cov}[X_{t-j}, X_{t-h-k}] = \sigma^2 \sum_{k \in \mathbb{Z}} \psi_k \psi_{k+h}.$$

This calculation is obtained by seeing that $\mathrm{Cov}[X_{t-j}, X_{t-h-k}]$ is zero unless $j = k + h$, in which case it equals σ^2.

Smoothing of Population

- We apply a simple moving average to U.S. Population, using the R *filter* function.
- Overlay different choices of m.

```
pop <- read.table("USpop.dat")
pop <- ts(pop, start = 1901)

m <- 20
ma.weights <- rep(1,2*m+1)/(2*m+1)
pop.trend1 <- stats::filter(pop,ma.weights,method="convolution",sides=2)

m <- 5
ma.weights <- rep(1,2*m+1)/(2*m+1)
pop.trend2 <- stats::filter(pop,ma.weights,method="convolution",sides=2)

plot(pop,xlab="Year",ylab="U.S. Population",col=1,lwd=1,yaxt="n",xaxt="n")
lines(ts(pop.trend1,start=1901,frequency=1),lty=1,lwd=1,col=2)
lines(ts(pop.trend2,start=1901,frequency=1),lty=1,lwd=1,col=3)
```



Lesson 3-3: Examples of Filters

- We explore smoothers, differencing, and the backshift operator.

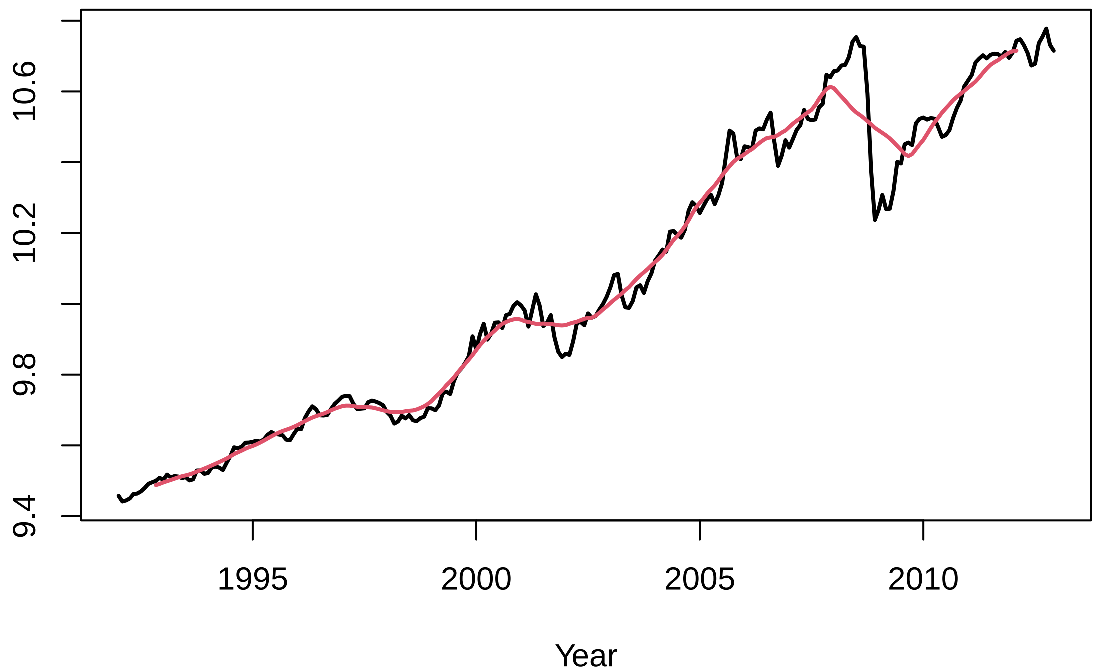
Paradigm 3.3.1. Smoother

- Linear filters that suppress oscillations and reveal long-term trends are called *smoothers*.
- A simple moving average is an example of a smoother.

Example 3.3.2. Smoothers Applied to Gasoline Sales

- We apply a simple moving average with $m = 10$ to the logged seasonally adjusted gasoline series.

```
gassa <- read.table("GasSA_2-11-13.dat")
gassa.log <- ts(log(gassa),start=1992,frequency=12)

h <- 10
simple.ma <- rep(1,2*h+1)/(2*h+1)
gas.trend <- stats::filter(gassa.log,simple.ma,method="convolution",sides=2)
gas.trend <- ts(gas.trend,start=1992,frequency=12)
plot(ts(gassa.log,start=1992,frequency=12),col=1,ylab="",xlab="Year",lwd=2)
lines(ts(gas.trend,start=1992,frequency=12),col=2,lwd=2)
```



- We can run a visualization in the notebook.

```
### Movie
movie <- FALSE
delay <- 0
flex <- 0
siglen <- length(gas.trend) - 2*h
range <- seq(1,siglen,1)
if(movie) {
for(t in range)
{
    Sys.sleep(delay)
    filterweight <- c(rep(NA,t-1),simple.ma + gas.trend[h+t] + flex,
        rep(NA,siglen-t))
    plot(ts(gassa.log,start=1992,frequency=12),col=1,ylab="",xlab="Year",lwd=2)
    lines(ts(gas.trend[1:(h+t)],start=1992,frequency=12),col=2,lwd=2)
    lines(ts(filterweight,start=1992,frequency=12),col="#00FF0090",lwd=2)
} }
```

Paradigm 3.3.3. Difference Filter

- To suppress long-term dynamics, we can *difference*:

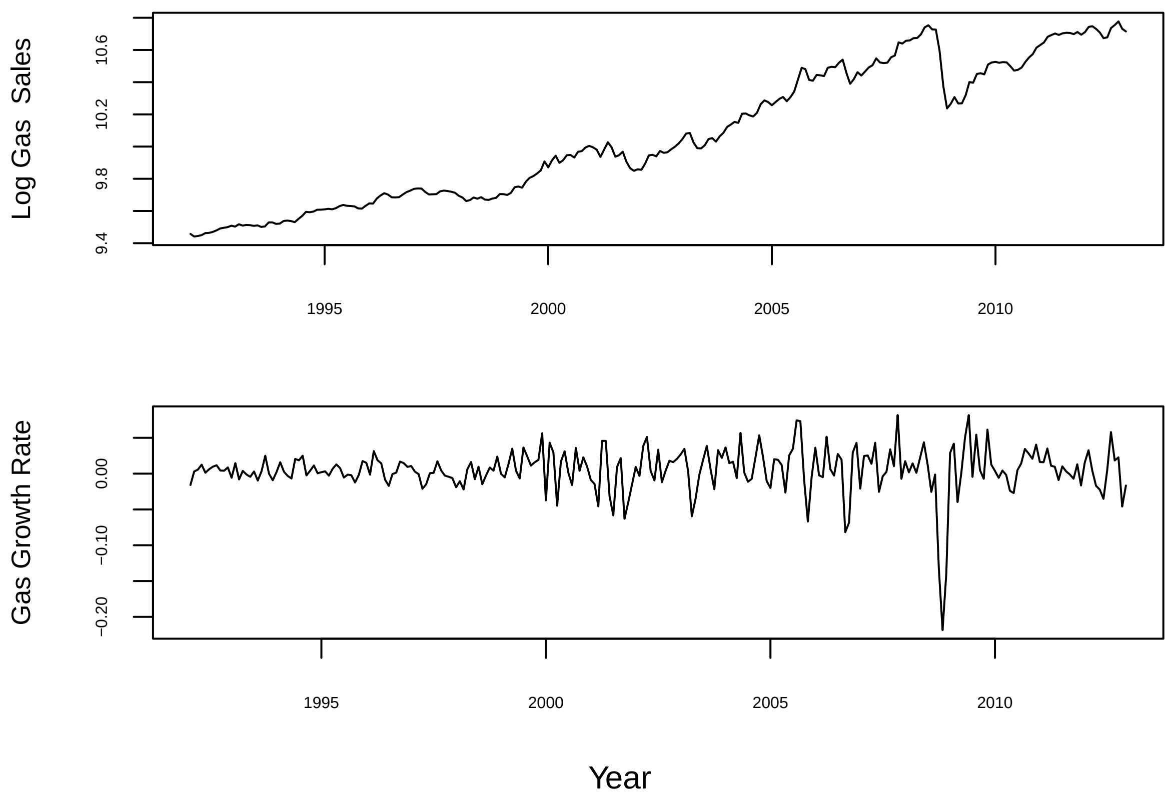
$$Y_t = X_t - X_{t-1}.$$

  This is a filter with weights $\psi_0 = 1$, $\psi_1 = -1$, and zero otherwise. It is called the *differencing filter*. This filter reduces polynomials in time by one degree; lines are reduced to constants.

Example 3.3.4. Differencing Applied to Gasoline Sales

- We apply differencing to logged seasonally adjusted gasoline sales. The result can be interpreted as a *growth rate*.

```
gas.diff <- diff(gassa.log)
par(oma=c(2,0,0,0),mar=c(2,4,2,2)+0.1,mfrow=c(2,1),cex.lab=.8)
plot(gassa.log,xlab="",ylab="Log Gas  Sales",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
plot(gas.diff,xlab="",ylab = "Gas Growth Rate",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
mtext(text="Year",side=1,line=1,outer=TRUE)
```



The Backward Shift

- We define the *backward shift* (or *lag*) operator B, which shifts a time series back one time unit. This is a linear filter that lags time by one unit:

$$Y_t = X_{t-1}.$$

  So the filter weights are $\psi_1 = 1$ and zero otherwise. Informally we write $BX_t = X_{t-1}$.

Powers of Backward Shift

- Then $X_{t-k} = B^k X_t$, and a general filter is expressed as

$$Y_t = \sum_{k \in \mathbb{Z}} \psi_k \, X_{t-k} = \sum_{k \in \mathbb{Z}} \psi_k B^k \, X_t,$$

so we write the filter as $\sum_{k \in \mathbb{Z}} \psi_k B^k$. Call this $\Psi(B)$. (Mathematically, $\Psi(z)$ is a Laurent series.)

Simple Moving Average Filter

- Expressed as $\Psi(B) = (2m+1)^{-1} \sum_{k=-m}^{m} B^k$.

Differencing Filter

- Expressed as $\Psi(B) = 1 - B$.

Lesson 3-4: Trends

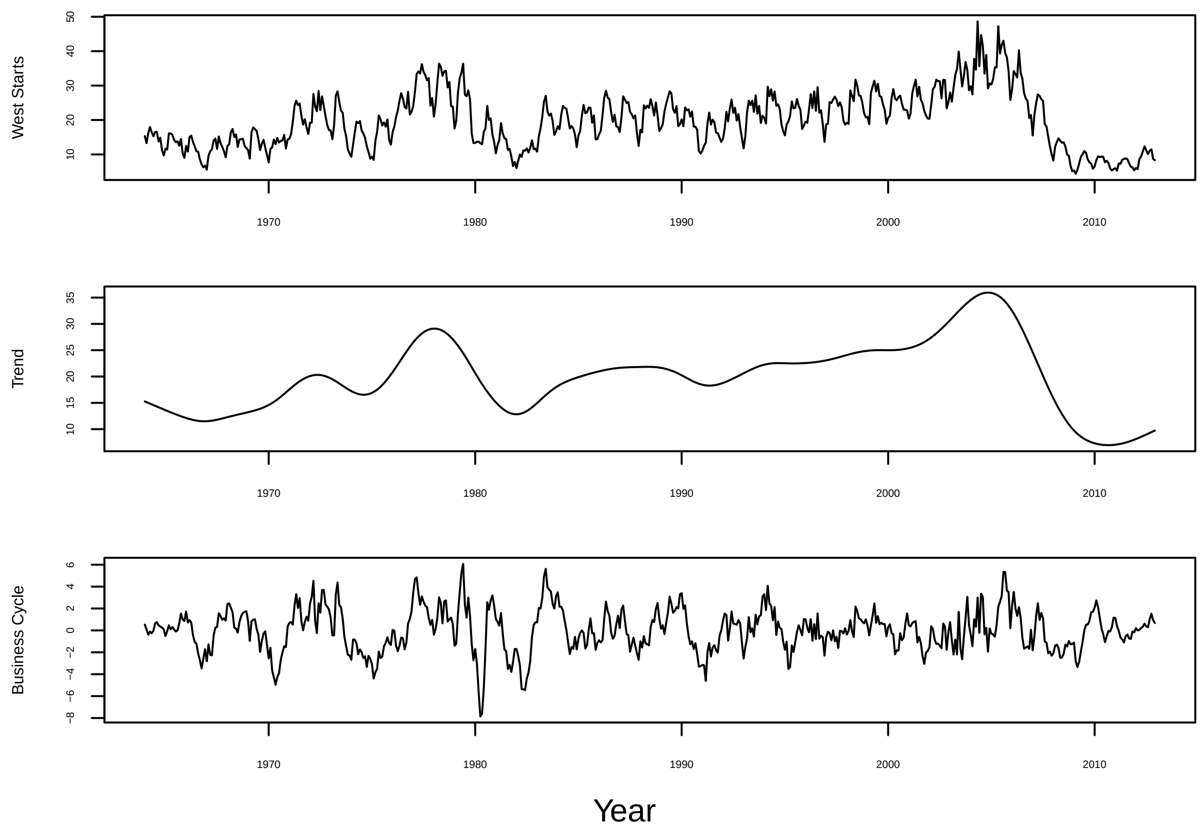
- Trends measure the long-term behavior of a time series.
- Trends are steady movements in the level of a time series.
- We can use smoothers to estimate trends.
- Removing a trend by subtraction allows us to see other features.

Example 3.4.1. Western Housing Starts Cycle

- We estimate and eliminate the trend in the Westing Housing Starts data, so that we can more clearly see the business cycle movements.

```
hpsa <- function(n,period,q,r)
{
    # hpsa
    #     gives an HP filter for seasonal data
    #   presumes trend+seas+irreg structure
    #       trend is integrated rw
    #       seas is seasonal rw
    #       irreg is wn
    #   q is snr for trend to irreg
    #   r is snr for seas to irreg

# define trend differencing matrix

delta.mat <- diag(n)
temp.mat <- 0*diag(n)
temp.mat[-1,-n] <- -2*diag(n-1)
delta.mat <- delta.mat + temp.mat
temp.mat <- 0*diag(n)
temp.mat[c(-1,-2),c(-n,-n+1)] <- 1*diag(n-2)
delta.mat <- delta.mat + temp.mat
diff.mat <- delta.mat[3:n,]

# define seasonal differencing matrix

delta.mat <- diag(n)
temp.mat <- 0*diag(n)
inds <- 0
for(t in 1:(period-1))
{
    temp.mat <- 0*diag(n)
    temp.mat[-(1+inds),-(n-inds)] <- 1*diag(n-t)
    delta.mat <- delta.mat + temp.mat
```

```
    inds <- c(inds,t)
}
sum.mat <- delta.mat[period:n,]

# define two-comp sig ex matrices

#trend.mat <- solve(diag(n) + t(diff.mat) %*% diff.mat/q)
#seas.mat <- solve(diag(n) + t(sum.mat) %*% sum.mat/r)
trend.mat <- diag(n) - t(diff.mat) %*% solve(q*diag(n-2) + diff.mat %*%
    t(diff.mat)) %*% diff.mat
seas.mat <- diag(n) - t(sum.mat) %*% solve(r*diag(n-period+1) + sum.mat %*%
    t(sum.mat)) %*% sum.mat

# define three-comp sig ex matrices

trend.filter <- solve(diag(n) - trend.mat %*% seas.mat) %*%
    trend.mat %*% (diag(n) - seas.mat)
seas.filter <- solve(diag(n) - seas.mat %*% trend.mat) %*%
    seas.mat %*% (diag(n) - trend.mat)
irreg.filter <- diag(n) - (trend.filter + seas.filter)

filters <- list(trend.filter,seas.filter,irreg.filter)
return(filters)
}

Wstarts <- read.table("Wstarts.b1",skip=2)[,2]
Wstarts <- ts(Wstarts,start = 1964,frequency=12)
n <- length(Wstarts)
q <- .0001
r <- 1
hp.filters <- hpsa(n,12,q,r)

wstarts.trend <- hp.filters[[1]] %*% Wstarts
wstarts.seas <- hp.filters[[2]] %*% Wstarts
wstarts.cycle <- hp.filters[[3]] %*% Wstarts
wstarts.sa <- wstarts.trend + wstarts.cycle

comps <- ts(cbind(wstarts.trend,wstarts.seas,wstarts.cycle),start=1964,frequency=12)
trend <- ts(wstarts.trend,start=1964,frequency=12)
cycle <- ts(wstarts.cycle,start=1964,frequency=12)

par(oma=c(2,0,0,0),mar=c(2,4,2,2)+0.1,mfrow=c(3,1),cex.lab=.8)
plot(Wstarts, ylab="West Starts",xlab="",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
plot(trend,xlab="",ylab = "Trend",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
plot(cycle,xlab="",ylab = "Business Cycle",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
mtext(text="Year",side=1,line=1,outer=TRUE)
```



Paradigm 3.4.12. Nonparametric Trend Estimation

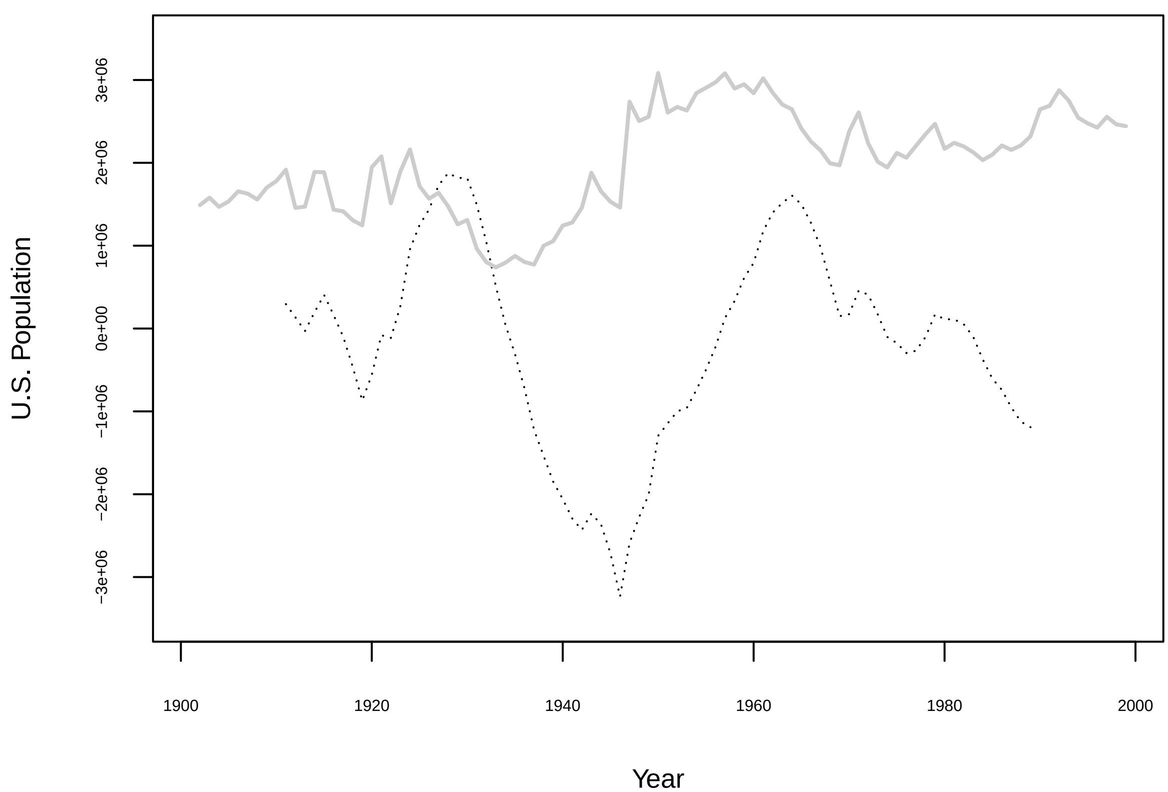
- We consider trends μ_t to be either a deterministic (but unknown) function of time, or are an unobserved stochastic process:
$$X_t = \mu_t + Z_t.$$
Here $\mathbb{E}[X_t] = \mu_t$ and $\{Z_t\}$ is a mean zero time series.
- A *two-sided moving average* is a linear filter that can be used to estimate trends:
$$\Psi(B) = \sum_{k=-m}^{m} \psi_k B^k.$$
When all the weights are equal, this is a simple moving average. It is two-sided because the moving average uses past and future data:
$$\widehat{\mu}_t = \Psi(B) X_t = \sum_{k=-m}^{m} \psi_k X_{t-k}.$$
- A *symmetric moving average* weights future and past equally, i.e., $\psi_k = \psi_{-k}$.
- As an estimator $\widehat{\mu}_t$ of the unknown trend μ_t, there is less bias if m is small, and less variance if m is large. This is a **Bias-Variance Dilemma**.

Paradigm 3.4.15. Trend Elimination

- We can remove a trend by subtracting off its estimate.
- We can also apply the differencing filter.

- Example with US population, detrended by a $m = 10$ simple moving average and by the difference filter.

```
pop <- read.table("USpop.dat")
pop <- ts(pop, start = 1901)

p <- 10
pop.smooth <- stats::filter(pop,rep(1,2*p+1)/(2*p+1),method="convolution")
pop.smooth <- ts(pop.smooth,start= 1901)
pop.diff <- diff(pop)

par(mar=c(4,4,2,2)+0.1,cex.lab=.8)
plot(ts(pop - pop.smooth,start=1901),lwd=1,lty=3,ylim=c(-3.5*10^6,3.5*10^6),
    ylab="U.S. Population",xlab="Year",yaxt="n",xaxt="n")
axis(1,cex.axis=.5)
axis(2,cex.axis=.5)
lines(pop.diff,col=gray(.8),lwd=2)
```



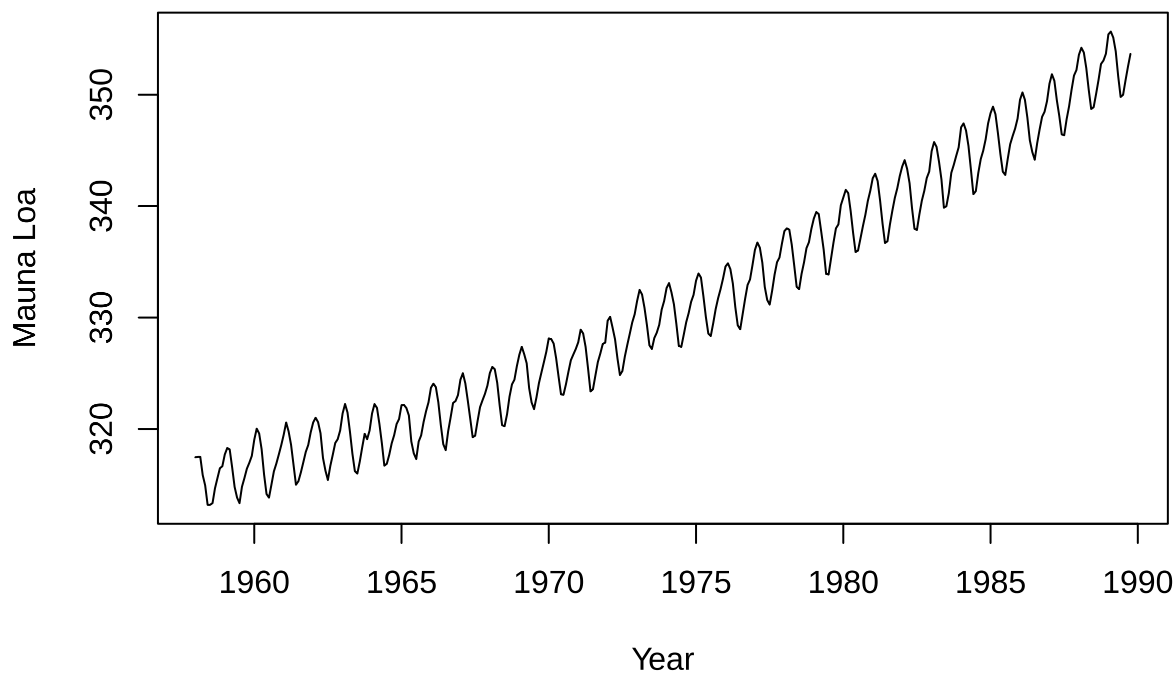
Lesson 3-5: Seasonality

- Regular patterns of a periodic nature are called *seasonality*.
- The same notion as a *cycle*, but is associated with the calendar.
- We may wish to estimate and remove seasonality in order to see other patterns.

Example 3.5.1. Mauna Loa Growth Rate

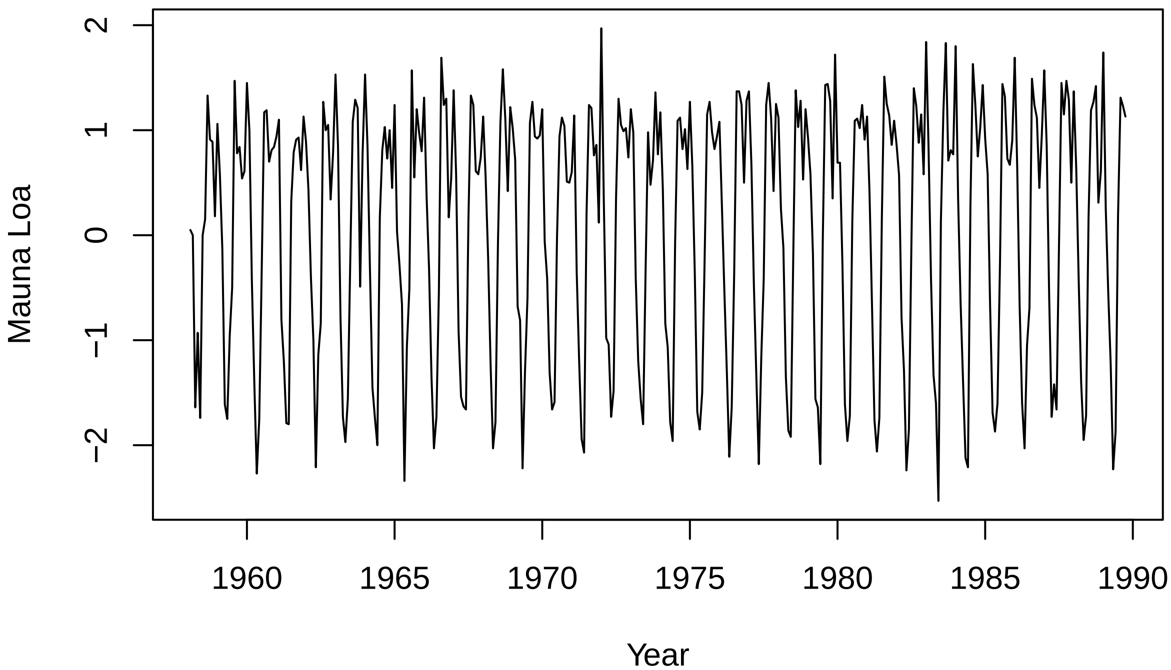
- We consider the Mauna Loa CO2 time series.

```
mau <- read.table("mauna.dat",header=TRUE,sep="")
mau <- ts(mau,start=1958,frequency=12)
plot(mau,xlab="Year",ylab="Mauna Loa")
```



- We can difference the series to eliminate the trend.
- A pronounced seasonal pattern is apparent; but it is not completely periodic.

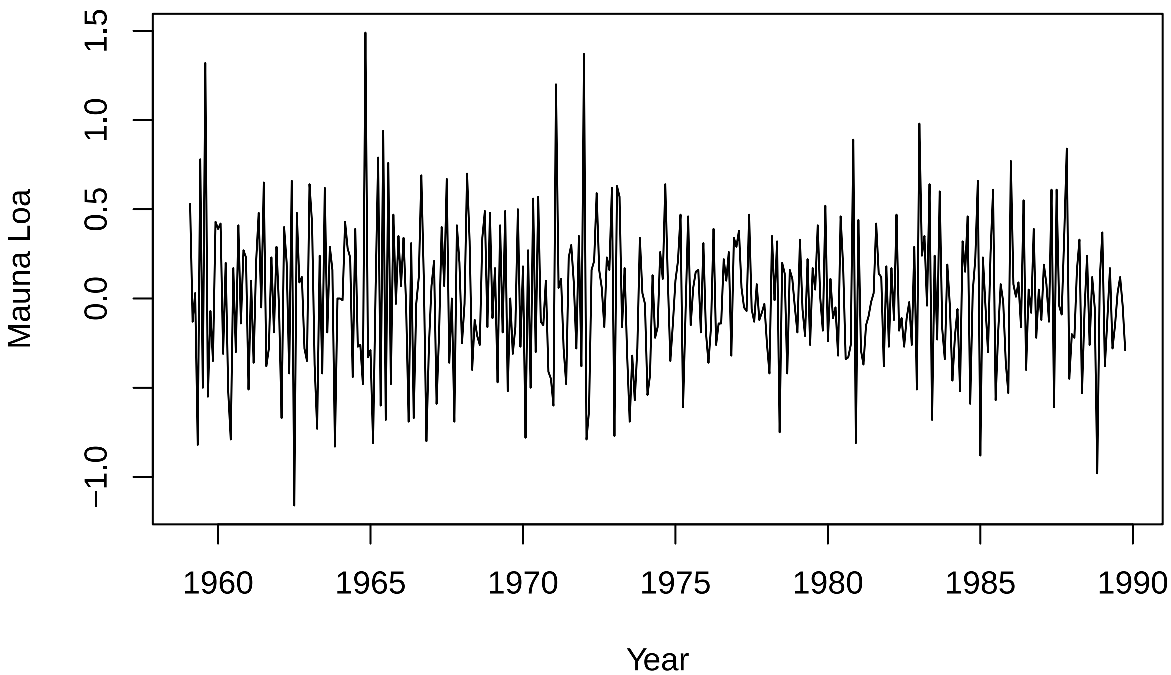
```
mau.diff <- diff(mau)
plot(mau.diff,xlab="Year",ylab="Mauna Loa")
```



Seasonal Regression

- An exactly periodic effect would be $X_t = X_{t-s}$ for seasonal period s (e.g., $s = 12$ for monthly time series), and any t.
- If this were true, we could use dummy regressors. Usually this works poorly; if the series were periodic, then the filter $1 - B^s$ would annihilate it.
- Example of Mauna Loa; seasonal differencing of the growth rate leaves a non-zero time series.

```
mau.diffs <- diff(mau.diff,lag=12)
plot(mau.diffs,xlab="Year",ylab="Mauna Loa")
```



Seasonal Moving Average

- A better way is to estimate seasonality using a *seasonal moving average*. This is a moving average filter where only coefficients with indices that are a multiple of s are non-zero.
- This means

$$\Psi(B) = \sum_{k=-ms}^{ms} \psi_k B^k = \sum_{j=-m}^{m} \psi_{js} B^{js}$$

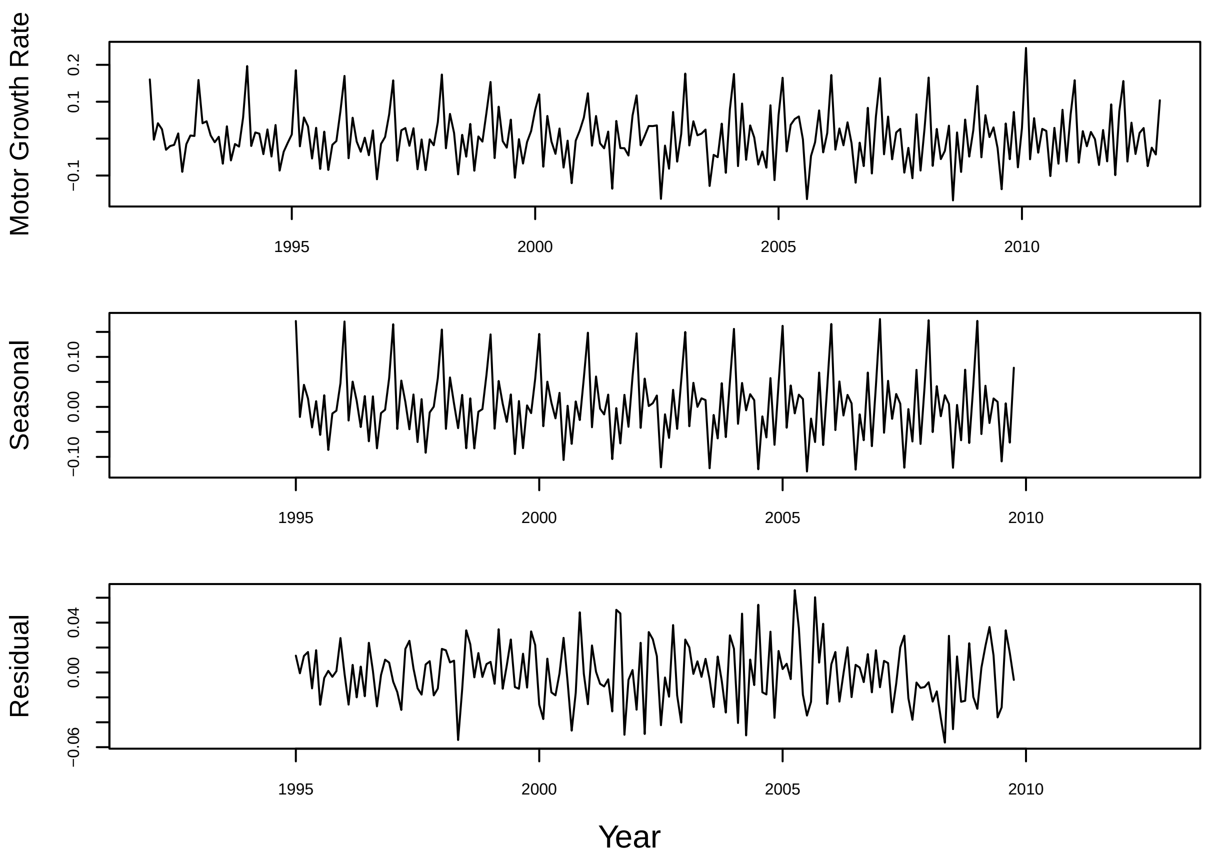
  is a seasonal moving average, because $\psi_k = 0$ unless $k = js$ for some integer j.

Example 3.5.8. Seasonal Moving Average for Motor Retail Sales

- We apply a seasonal moving average to the growth rate of the Motor Retail Sales data, with $m = 3$.
- We plot the growth rate series, the estimated seasonality, and the residual after subtracting off the seasonal component.

```
Ret441 <- read.table("retail441.b1",header=TRUE,skip=2)[,2]
Ret441 <- ts(Ret441,start = 1992,frequency=12)
ret.diff <- diff(log(Ret441))

pseas <- 3
period <- 12
seas.smoother <- c(1,rep(0,period-1))
seas.smoother <- c(rep(seas.smoother,pseas),rep(seas.smoother,pseas),1)/(2*pseas+1)
seas.smooth <- stats::filter(ret.diff,seas.smoother,method="convolution")
seas.resid <- ret.diff - seas.smooth
```

```r
par(oma=c(2,0,0,0),mar=c(2,4,2,2)+0.1,mfrow=c(3,1),cex.lab=1.2)
plot(ret.diff,ylab="Motor Growth Rate",xlab="",yaxt="n",xaxt="n")
axis(1,cex.axis=.8)
axis(2,cex.axis=.8)
plot(ts(seas.smooth,start=1992,frequency=12),xlab="",ylab="Seasonal",yaxt="n",xaxt="n")
axis(1,cex.axis=.8)
axis(2,cex.axis=.8)
plot(ts(seas.resid,start=1992,frequency=12),xlab="",ylab="Residual",yaxt="n",xaxt="n")
axis(1,cex.axis=.8)
axis(2,cex.axis=.8)
mtext(text="Year",side=1,line=1,outer=TRUE)
```



Remark 3.5.9. Seasonal Adjustment

- Removal of seasonality is called *seasonal adjustment*.
- The *seasonal aggregation* filter is defined as

$$U(B) = 1 + B + B^2 + \ldots + B^{s-1}.$$

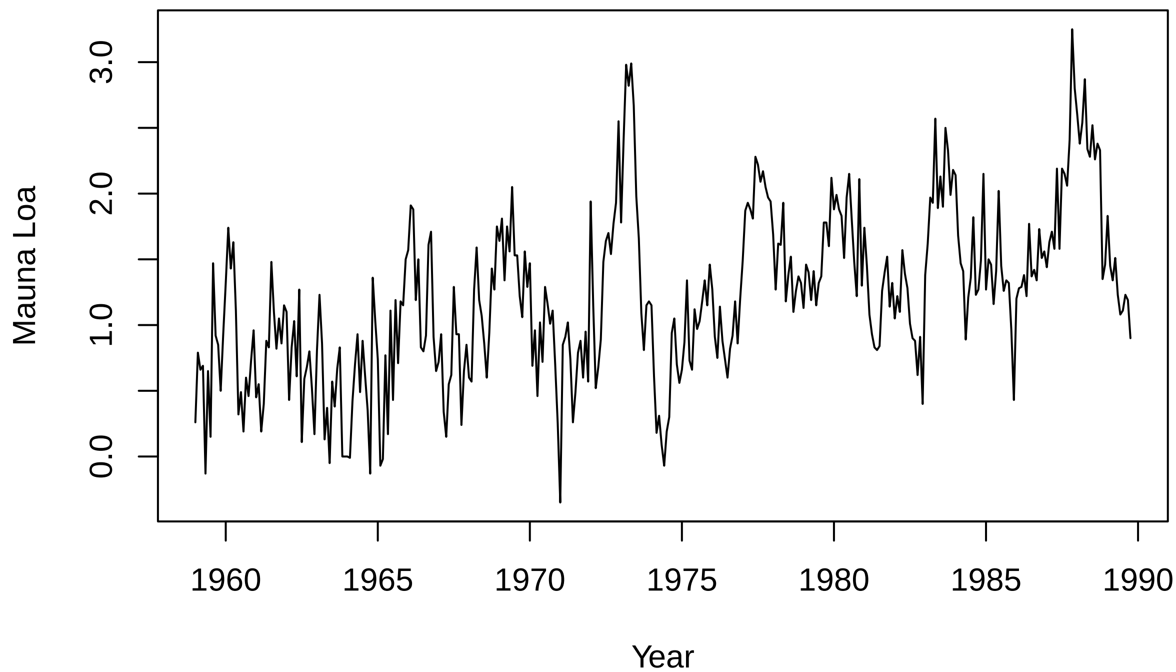
  It annihilates any periodic pattern, and is a basic example of a seasonal adjustment filter.

Example 3.5.10. Seasonal Adjustment of Mauna Loa Growth Rate

- We apply the seasonal aggregation filter to the growth rate of the Mauna Loa CO2 time series.
- This uses a trick: applying $U(B)$ and $1 - B$ together amounts to applying $1 - B^s$, because

$$U(B)(1 - B) = 1 - B^s.$$

```r
seas.elim <- diff(mau,lag=12)
plot(seas.elim,xlab="Year",ylab="Mauna Loa")
```



Lesson 3-6: Trend and Seasonality Together

- We may want to extract both trend and seasonality from time series.
- This is more challenging than just extracting trend or seasonality alone.

Proposition 3.6.5

- Stable seasonality has an exactly periodic pattern.
- A trend extraction filter $\Psi(B)$ eliminates stable seasonality if it can be written as

$$\Psi(B) = \Theta(B)U(B),$$

  for some filter $\Theta(B)$, where $U(B)$ is the seasonal aggregation filter.

Example 3.6.7. Trend Filters that Eliminate Seasonality

- For s even, consider the trend filter

$$\Psi(B) = \frac{1}{2s}B^{-s/2} + \frac{1}{s}B^{-s/2+1} + \ldots + \frac{1}{s}B^{s/2-1} + \frac{1}{2s}B^{s/2}.$$

- By polynomial multiplication, we can show that

$$\Psi(B) = \frac{1}{2s}U(B)(1+B)B^{-s/2}.$$

- Setting $\Theta(B) = (1+B)B^{-s/2}/(2s)$, we see that $\Psi(B)$ eliminates stable seasonality.

Paradigm 3.6.8. Classical Decomposition from Trend Filtering

- Consider a time series with both trend and seasonal effects.
- We can design a trend filter that eliminates seasonality, obtaining $\widehat{T}_t$.
- Then apply a seasonal filter to $X_t - \widehat{T}_t$, the de-trended data, obtaining $\widehat{S}_t$.
- Removing $\widehat{S}_t$ from the de-trended data yields the irregular $\widehat{I}_t$.
- Then we have the *classical decomposition*

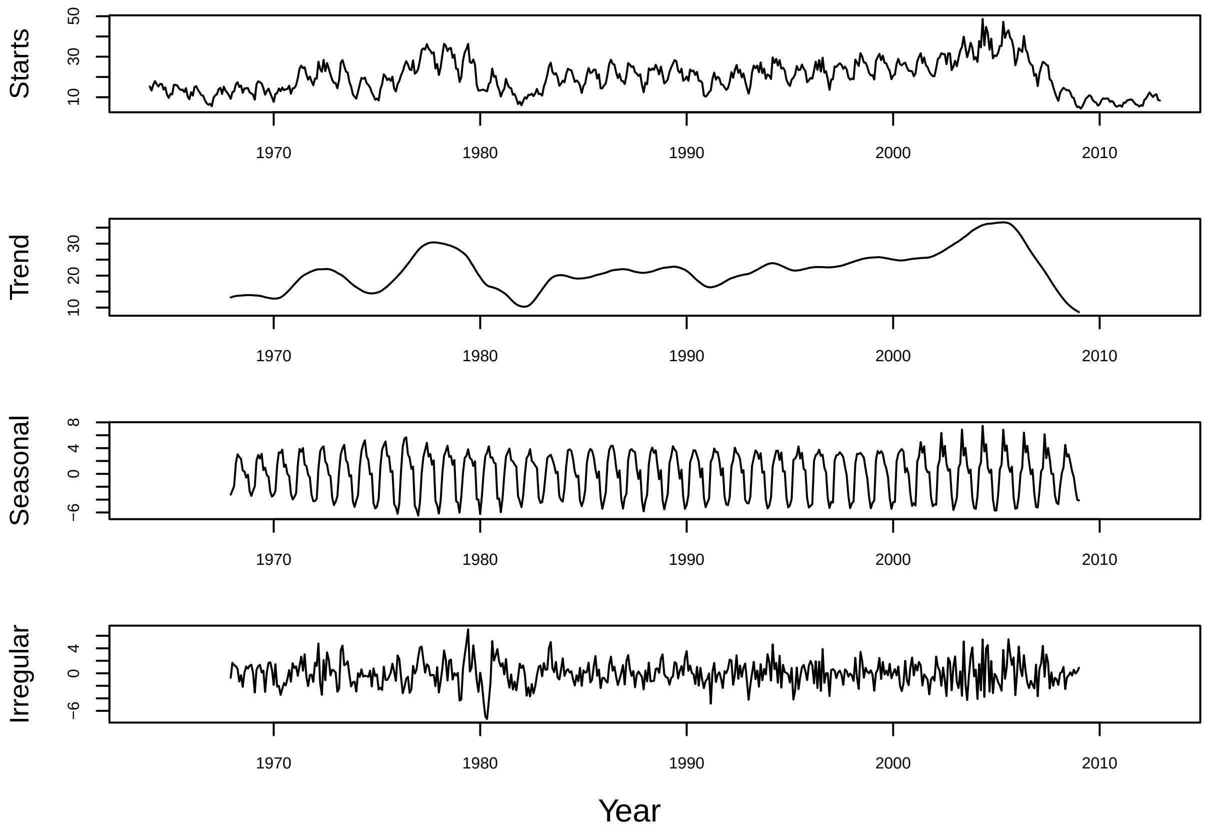
$$X_t = \widehat{T}_t + \widehat{S}_t + \widehat{I}_t.$$

Example 3.6.13. Classical Decomposition of Western Housing Starts via Linear Filtering

- Consider the trend filter given in Example 3.6.7.
- We construct the classical decomposition, applied to Western Housing Starts

```
compSmooth <- function(data,period,pseas)
{
    # compSmooth
    #   Estimates trend, seasonal, and irregular from input data
    #   using weighted moving averages.
    #   data: an input time series
    #   period: observations per year
    #   pseas: odd integer giving number of years in seasonal moving average
    #

    n <- length(data)
    trendfilter <- seq(1,period)
    trendfilter <- c(trendfilter[1:(period-1)],rev(trendfilter))/period^2
    seasfilter <- c(1,rep(0,period-1))
    seasfilter <- c(rep(seasfilter,pseas),rep(seasfilter,pseas),1)/(2*pseas+1)
    trend <- stats::filter(data,trendfilter,method="convolution",sides=2)
    trend <- trend[period:(n-period+1)]
    resid <- data[period:(n-period+1)] - trend
    seasonal <- stats::filter(resid,seasfilter,method="convolution",sides=2)
    seasonal <- seasonal[(pseas*period + 1):(length(resid) - pseas*period)]
    irreg <- resid[(pseas*period + 1):(length(resid) - pseas*period)] - seasonal

    return(cbind(trend[(pseas*period + 1):(length(resid) - pseas*period)],seasonal,irreg))
}

Wstarts <- read.table("Wstarts.b1",skip=2)[,2]
Wstarts <- ts(Wstarts,start = 1964,frequency=12)
pseas <- 3
comps <- compSmooth(Wstarts,12,pseas)

par(oma=c(2,0,0,0),mar=c(2,4,2,2)+0.1,mfrow=c(4,1),cex.lab=1.2)
plot(Wstarts,ylab="Starts",xlab="",yaxt="n",xaxt="n")
axis(1,cex.axis=.8)
axis(2,cex.axis=.8)
plot(ts(c(rep(NA,47),comps[,1],rep(NA,47)),start=1964,frequency=12),xlab="",ylab="Trend",yaxt="n",xaxt=
axis(1,cex.axis=.8)
axis(2,cex.axis=.8)
```

```
plot(ts(c(rep(NA,47),comps[,2],rep(NA,47)),start=1964,frequency=12),xlab="",ylab="Seasonal",yaxt="n",xa
axis(1,cex.axis=.8)
axis(2,cex.axis=.8)
plot(ts(c(rep(NA,47),comps[,3],rep(NA,47)),start=1964,frequency=12),xlab="",ylab="Irregular",yaxt="n",xa
axis(1,cex.axis=.8)
axis(2,cex.axis=.8)
mtext(text="Year",side=1,line=1,outer=TRUE)
```



Lesson 3-7: Integrated Processes

- Sometimes non-stationarity arises in the mean function $\mathbb{E}[X_t] = \mu_t$.
- But the mean may not explain all the non-stationarity of a time series.
- *Integrated Processes* are stochastic processes that exhibit a type of non-stationarity.

Example 3.7.1. Random Walk

- We revisit the *random walk* $\{X_t\}$, where

$$X_t = X_{t-1} + Z_t$$

  for $t \geq 1$, and $\{Z_t\}$ are i.i.d. $(0, \sigma^2)$.

- We can initialize with $X_0 = 0$.
- Note that differencing yields the increment:

$$(1 - B)X_t = Z_t.$$

- Writing the recursion out yields

$$X_t = X_0 + \sum_{k=1}^{t} Z_k.$$

- This is a cumulation, hence an *integrated process*.

Integrated Process

- Suppose that $\{X_t\}$ is a non-stationary stochastic process such that

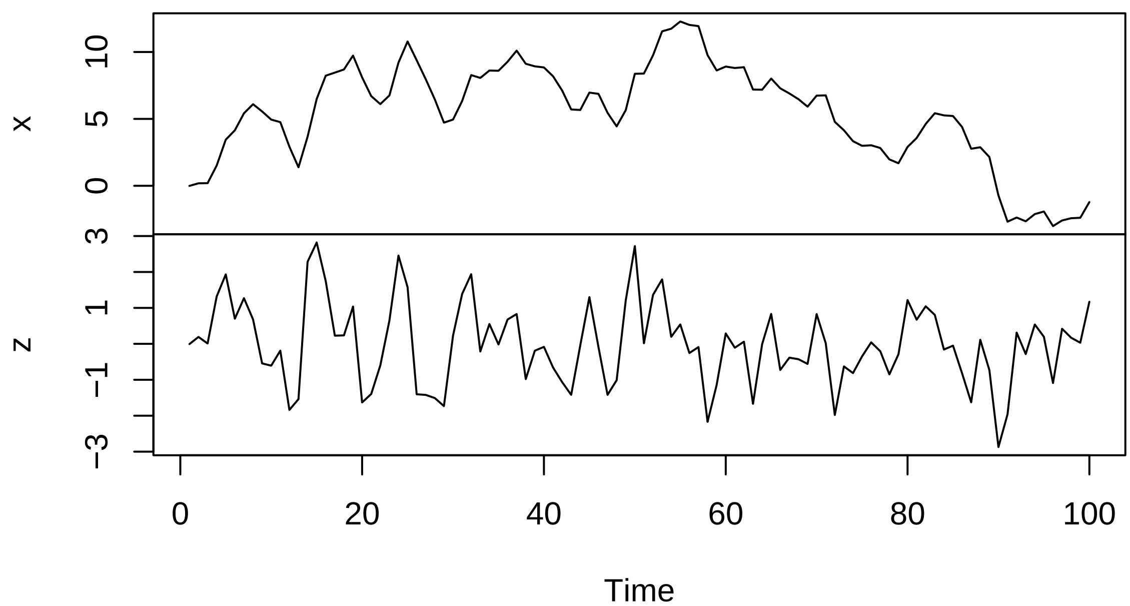
$$(1 - B)X_t = Z_t$$

is stationary. Then it is called an *integrated process*.
- Differencing $1 - B$ cancels out the cumulation.

Example of Integrated Moving Average

- Suppose that $\{Z_t\}$ is an MA(1), and $X_t = X_{t-1} + Z_t$.
- We simulate a sample path.

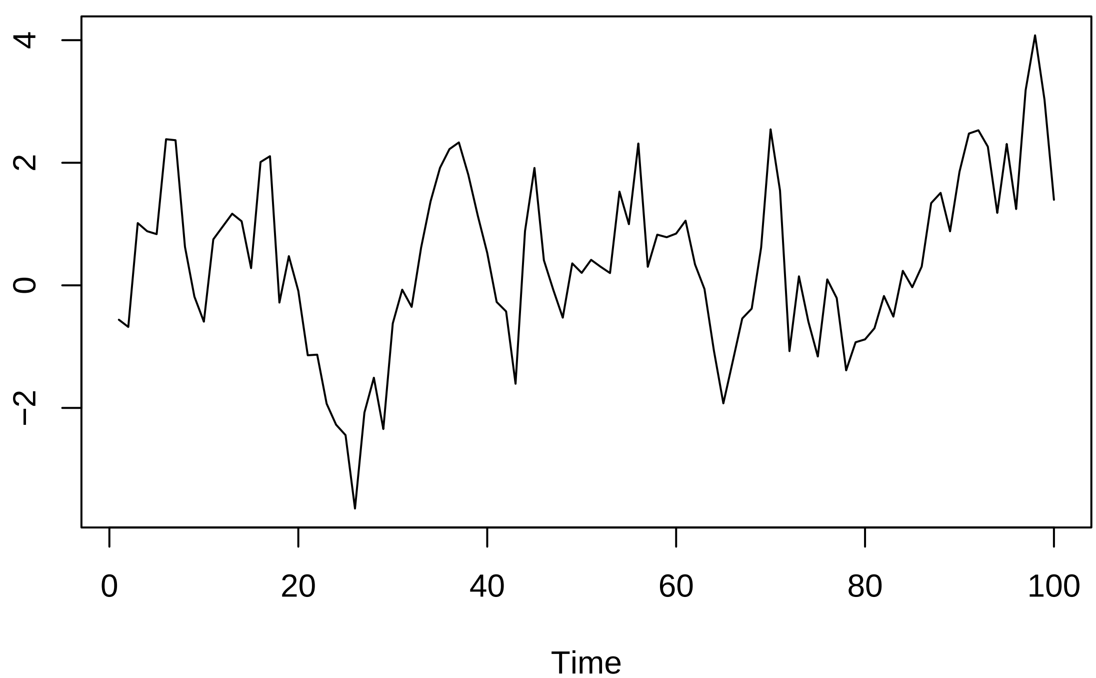
```
set.seed(777)
n <- 100
eps <- rnorm(n+1)        # Gaussian input
theta <- .8
z <- eps[-1] + theta*eps[-(n+1)]
x <- cumsum(z)
plot(ts(cbind(x,z)),xlab="Time",ylab="",main="")
```

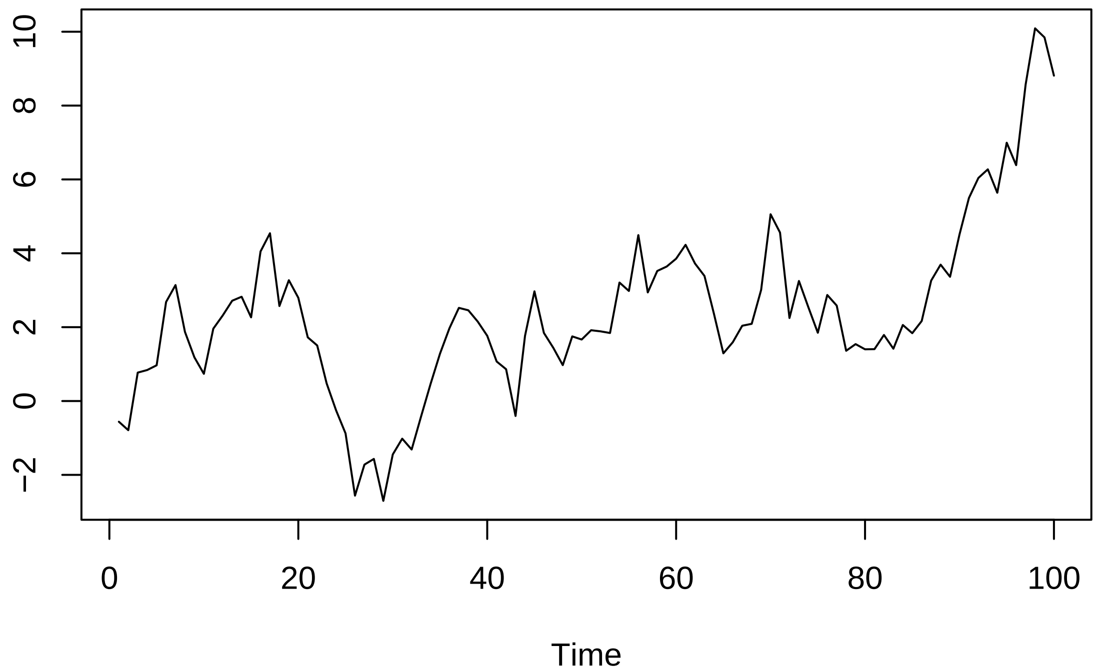


Example 3.7.3. Unit Root Process

- The AR(1) process $\{X_t\}$ satisfies
$$X_t = \phi X_{t-1} + Z_t,$$
and can be written as
$$(1 - \phi B)X_t = Z_t.$$
So when $\phi = 1$, the AR(1) is a random walk. The polynomial $1 - \phi z$ has root $1/\phi$, and when this root equals one, the root is *unit*. Hence a random walk (and more generally, an integrated process) is sometimes called a *unit root process*.
- Consider a sample path of an AR(1) with high value of ϕ.

```
n <- 100
for(i in 1:2)
{
set.seed(123)
z <- rnorm(n)
x <- rep(0,n)
phi <- .80+(i-1)*.199
x0 <- 0
x[1] <- x0 + z[1]
for(t in 2:n) { x[t] <- phi*x[t-1] + z[t] }
plot(ts(x),xlab="Time",ylab="")
}
```




10
8
6
4
2
0
−2
0
20
40
60
80
100
Time