SINGULAR SPECTRUM ANALYSIS IN R

JAN DE LEEUW AND PATRICK CRUTCHER

ABSTRACT. Meet the abstract. This is the abstract.

1. INTRODUCTION

Singular Spectrum Analysis (SSA from now on) decomposes an observed time series into a sum of component series, in which the components hopefully capture and show the dynamics of the series more clearly. The SSA decomposition method can be thought of as one possible generalization of the singular value decomposition (SVD), or of principal component analysis (PCA), to a single time series. It can display and isolate trends and seasonal effects, as well as stationary residuals.

SSA originated and has mostly been studied and applied in geophysics and atmospheric science. A comprehensive early review paper is Ghil et al. [2002]. The technique is not very well known is statistics, despite the publication some time ago of the book by Golyandina et al. [2001]. Recent examples, perhaps showing an increasing interest, are the review paper by Hassani [2007], and the application published by Bilancia and Stea [2008].

2. PCA OF STATIONARY SERIES

Suppose $\underline{x}$ is a T-dimensional vector random variable with $\mathbf{E}(\underline{x}\underline{x}') = \Omega$. The cross product matrix Ω, which we assume to be positive definite, has spectral decomposition $\Omega = Q\Lambda^2 Q'$. Define $\underline{z} = \Lambda^{-1}Q'\underline{x}$. Then $\mathbf{E}(\underline{z}\underline{z}') =$

Date: September 24, 2009 — 14h 45min — Typeset in LUCIDA BRIGHT.

2000 *Mathematics Subject Classification.* 62M10.

Key words and phrases. Time Series, Singular Value Decomposition, Singular Spectrum Analysis.

$\Lambda^{-1}Q'\Omega Q\Lambda^{-1} = I$. Also $\underline{x} = Q\Lambda\underline{z}$, the *Karhunen-Loève Decomposition* of $\underline{x}$.

In the usual time series analysis we only observe a single T-dimensional realization x of $\underline{x}$, and we have no idea what Ω is. If we assume $\underline{x}$ is stationary, however, we can estimate ω_{st} with $s \geq t$ by the average of the $T-(s-t)$ products $x_u x_v$ for which $u - v = s - t$. Thus

$$\hat{\omega}_{st} = \frac{1}{T-(s-t)} \sum_{v=1}^{T-(s-t)} x_v x_{v+(s-t)}.$$

Now compute $\hat{\Omega} = \hat{Q}\hat{\Lambda}^2\hat{Q}'$ and $\hat{z} = \hat{\Lambda}^{-1}\hat{Q}'x$. Again $x = \hat{Q}\hat{\Lambda}\hat{z}$. The columns of $\hat{Q}$ are *Empirical Orthogonal Functions* or EOF's.

More specific assumptions about the nature of the stationary process may lead to more precise estimates of Ω, provided of course these assumptions are more or less true. For instance, we might assume that $\underline{x}$ is **AR**(1), in which case Ω only depends on the variance σ^2 and the autocorrelation ρ.

```
dg<-function(s) {
    n<-nrow(s); nn<-1:n; r<-rep(0,n)
    for (k in 0:(n-1))
         r[k+1]<-mean(s[which(outer(nn,k+nn,"=="))])
    return(toeplitz(r))
}
```

3. The Four-Step Program

3.1. **Embedding.** Suppose $\{x_1, x_2, \cdots, x_T\}$ is our observed time series. For now, we assume there are no gaps (missing data). Choose a *window width* $2 \leq L \leq \lfloor \frac{1}{2}T \rfloor$. Define $K = T - L + 1$, so that $K = \lceil \frac{1}{2}T \rceil + 1 \geq L + 1$. Define the $K \times L$ matrix Z with row i equal to $\{x_i, x_{i+1}, \cdots, x_{i+L-1}\}$. Thus $z_{ij} = x_{i+j-1}$, which implies that Z is a *Hankel matrix*. Z is constant along its skew-diagonals, if $i + j = k + \ell$ then $z_{ij} = z_{k\ell}$.

3.2. **Decomposition.** The singular value decomposition or SVD of Z is $Z = U\Lambda V'$. Here U is a $K \times L$ orthonormal matrix, V is $L \times L$ square orthonormal, and Λ is a diagonal matrix of order L. We suppose, for

identification purposes, that the diagonal elements of Λ are non-negative and are in non-increasing order along the diagonal. The SVD can also be written as

$$Z = \sum_{s=1}^{L} Z_s = \sum_{s=1}^{L} \lambda_s u_s v_s'.$$

Each of the Z_s is a rank-one matrix, and the Z_s are orthogonal in the sense that both $Z_s Z_t' = 0$ and $Z_s' Z_t = 0$ for $s \neq t$. Also, for the sum of squares, a.k.a. the squared *Frobenius norm*,

$$\mathbf{SSQ}(Z) = \sum_{s=1}^{L} \mathbf{SSQ}(Z_s) = \sum_{s=1}^{L} \lambda_s^2,$$

If we define

$$\pi_t \stackrel{\Delta}{=} \frac{\lambda_t^2}{\sum_{s=1}^{L} \lambda_s^2}$$

then the π_t add up to one, and can be interpreted as the percentage of the sum of squares "explained" by Z_t, i.e. by the singular triple (λ_t, u_t, v_t).

3.3. **Grouping.** Suppose $\mathcal{I} = \{I_1, \cdots, I_\Xi\}$ is a *partition* of $\{1, 2, \cdots, L\}$, and define $\overline{Z}_\xi = \sum_{s \in I_\xi} Z_s$. Obviously we still have $Z = \sum_{\xi=1}^{\Xi} \overline{Z}_\xi$. Moreover $Z_\xi Z_\mu' = 0$ and $Z_\xi' Z_\mu = 0$ for $\xi \neq \mu$, and

$$\mathbf{SSQ}(Z) = \sum_{\xi=i}^{\Xi} \mathbf{SSQ}(\overline{Z}_\xi) = \sum_{\xi=1}^{\Xi} \left\{ \sum_{s \in I_\xi} \lambda_s^2 \right\}.$$

3.4. **Hankelization.** Suppose we have a partition $\mathcal{I}$ into Ξ sets, and corresponding $\overline{Z}_\xi$. Find the Hankel matrices $\tilde{Z}_\xi$ that minimize $\mathbf{SSQ}(Z - \overline{Z}_\xi)$. They can be computed simply by *Hankelizing* or *diagonal averaging*, i.e. by replacing all elements for which $i + j$ is constant by their average. At the same time this define a time series $\tilde{x}_\xi$, with element k equal to the elements of $\tilde{Z}_\xi$ for which $i + j = k - 1$. Since Hankelizing is a linear operation, and since Z is Hankel already, we have

$$Z = \sum_{\xi=1}^{\Xi} \tilde{Z}_\xi,$$

as well as

$$x = \sum_{\xi=1}^{\Xi} \tilde{x}_\xi.$$

This is the SSA decomposition of x, or rather it is *a* SSA decomposition, because it depends on the choice of the window width and the grouping of the singular triples.

Note, by the way, that Hankelizing a matrix is an orthogonal projection, and thus

$$\mathbf{SSQ}(Z) \geq \sum_{\xi=1}^{\Xi} \mathbf{SSQ}(\tilde{Z}_\xi),$$

as well as

$$\mathbf{SSQ}(x) \geq \sum_{\xi=1}^{\Xi} \mathbf{SSQ}(\tilde{x}_\xi).$$

4. Choices

4.1. **Window width.** There are many sophisticated methods to choose window width. Golyandina et al. [2001, p. 18] suggest determining the *fractal dimension* of the series, or to find an approximate order using *autoregression*. In our code we use $L = \lfloor \frac{1}{2}T \rfloor$ as the default, and generally that seems to work rather well.

4.2. **Grouping.** We have chosen a simple grouping method based on *w-correlations* defined in Golyandina et al. [2001, p. 46–47]. These are just cosines between time series, using a weighted inner product that de-emphasizes the beginning and end of the series.

We first use the trivial grouping in which each singular triple defines a group. Compute the $L \times L$ matrix R of w-correlations, and choose a cut-off quantity $0 < \epsilon < 1$. the adjacency matrix A defined by

$$a_{j\ell} = \begin{cases} 1 & \text{if } |r_{j\ell}| > \epsilon, \\ 0 & \text{otherwise.} \end{cases}$$

We then use *Warshall's Algorithm* [Warshall, 1962] to compute the *transitive closure* of the relation corresponding with A, and we use the *equivalence classes* of this relation to define the groups.

In principle any clustering method can be used here. In Bilancia and Stea [2008], for example, complete linkage hierarchical clustering is used. Given the almost infinite number of cluster methods that are available

there is a great deal of flexibility here [Gan et al., 2007] . Even if we limit ourselves to cluster methods available in R, we have many different choices. Also note that there are many clustering methods that do not require pairwise similarity measures first, and work directly on the data matrix.

5. Examples

5.1. **Nile.** Data are from the package `datasets` in base R. The series, of length 100, is the annual flow of the river Nile at Ashwan 1871–1970. A timeplot is given in Figure 1(a).

Insert Figure 1(a) *about here*

For the SSA we use window width 50 and cut-off 0.25. The first singular triple explains 97.4% and the next two 0.32% and 0.22%. The grouping gives the six groups $\{\{1\},\{2\},\{\cdots \text{rest} \cdots\},\{32,33\},\{36,37\},\{48,49\}\}$.

Insert Figure 1(b) *about here*

5.2. **Accidental Deaths.** Data are from package MASS and give monthly totals of accidental deaths in the USA from 1973–1978. The series has length 72, and is plotted in Figure 2(a).

Insert Figure 2(a) *about here*

For the SSA we use window width 36 and cut-off 0.25. The first singular triple explains 99.01% and the next two 0.68% and 0.12%. The grouping gives the ten groups

$$\{\{1\},\{2,3\},\{4,5\},\{6\},\{7,8\},\{9,10\},\{11,\cdots,18\},\{\cdots \text{rest} \cdots\},\{31\},\{35,36\}\}.$$

Insert Figure 2(b) *about here*

5.3. **Milk.** The data from package TSA give the average monthly milk production per cow in the US from 1994 to 2005. The series has length 144, and is plotted in Figure 3(a).

Insert Figure 3(a) *about here*

SSA with the default settings gives 10 components. The first component explains 99.85% of the total sum of squares. The groups are

$$\{\{1\},\{2,3\},\{4,5\},\{6\},\{7,8\},\{\cdots\text{rest}\cdots\},$$
$$\{12,13\},\{34,35\},\{44,\cdots,70\},\{71,72\}\}.$$

The 10 components are plotted in Figure 3(b).

Insert Figure 3(b) *about here*

5.4. **Beer Sales.** Data from package TSA. Monthly beer sales in millions of barrels from 1975 to 1990. The series has length 192, and is plotted in Figure 4(a).

Insert Figure 4(a) *about here*

Insert Figure 4(b) *about here*

5.5. **CO_2.** Data from package `datasets`. Mauna Loa Atmospheric CO_2 Concentration, monthly 1959–1997. The series has length 468, and is plotted in Figure 5.

Insert Figure 5 *about here*

Window width is 234, and the algorithm with cut-off 0.25 gives 12 groups. The first eigenvalue explains 99.9955% of the total sum of squares.

6. Variations

6.1. **To Center or Not to Center.**

6.2. **From Hankel to Toeplitz.** Expanding the time series to a Hankel matrix, and decomposing this matrix into a sum of Hankel matrices, takes up a large amount of space. We can avoid all this, by working with the matrix $C = Z'Z$ and its eigen-decomposition $C = V\Lambda^2V'$. Now

$Z_s = Z v_s v_s'$, which becomes in elementwise notation

$$(Z_s)_{ik} = \sum_{j=1}^{L} z_{ij} v_{js} v_{ks} = \sum_{j=1}^{L} x_{i+j-1} v_{js} v_{ks}$$

Hankelizing means setting

$$(x_s)_\nu = \frac{1}{n_\nu} \sum \{(Z_s)_{ik} \mid i + k = \nu + 1\}$$

Suppose (i, k) are the n_ν index pairs for which $1 \le i \le K$ and $1 \le k \le L$ and $i + k = \nu + 1$. This means that $\max(1, (\nu + 1) - L) \le i \le \min(\nu, K)$ and $k = (\nu + 1) - i$. Thus

$$(x_s)_\nu = \frac{1}{n_\nu} \sum_{i=\max(1,(\nu+1)-L)}^{\min(\nu,K)} \sum_{j=1}^{L} x_{i+j-1} v_{js} v_{(\nu+1)-i,s}$$

The expression only involves the eigenvectors of C and the values of the original series. No *Embedding* and no $Hankelizing$ is required.

6.3. **Effect of Window Width.**

7. Extensions

7.1. **Gaps.**

7.2. **Multivariate Series.**

References

M. Bilancia and G. Stea. Timescale Effect Estimation in Time-series Studies of Air Pollution and Health: A Singular Spectrum Analysis Approach. *Electronic Journal of Statistics*, 2:432–453, 2008. URL `http://www.i-journals.org/ejs/include/getdoc.php?id=967&article=123&mode=pdf`.

G. Gan, C. Ma, and J. Wu. *Data Clustering. Theory, Algorithms, and Applications*. ASA-SIAM Series on Statistics and Applied Probability. SIAM, Philadelphia, PA, 2007.

M. Ghil, M.R. Allen, M.D. Dettinger, K. Ide, D. Kpondrashov, M.E. Mann, A.W. Robertson, A. Saunders, Y. Tian, F. Varadi, and P. Yiou. Advanced Spectral Methods for Climatic Time Series. *Review of Geophysics*, 40: 1–41, 2002.

N. Golyandina, V. Nekrutkin, and A. Zhigljavsky. *Analysis of Time Series Structure: SSA and Related Techniques*. Number 90 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, Florida, 2001.

H. Hassani. Singular Spectrum Analysis: Methodology and Comparison. *Journal of Data Science*, 5:239–257, 2007. URL `http://www.sinica.edu.tw/~jds/JDS-396.pdf`.

S. Warshall. A Theorem on Boolean Matrices. *Journal of the Association of Computer Machinery*, 9:11–12, 1962.

APPENDIX A. CODE

A.1. R Code.

```
#
#   ssa package
#   Copyright (C) 2008  Jan de Leeuw <deleeuw@stat.ucla.edu>
#   UCLA Department of Statistics, Box 951554,
#   Los Angeles, CA 90095-1554
#
#   This program is free software; you can redistribute it
#   and/or modify it under the terms of the GNU General Public
#   License as published by the Free Software Foundation;
#   either version 2 of the License, or (at your option)
#   any later version.
#
#   This program is distributed in the hope that it will be
#   useful, but WITHOUT ANY WARRANTY; without even the implied
#   warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
#   PURPOSE.  See the GNU General Public License for more details.
#
#   You should have received a copy of the GNU General Public
#   License along with this program; if not, write to the
#   Free Software Foundation, Inc., 675 Mass Ave, Cambridge,
#   MA 02139, USA.
#
######################################################################
#
# version 0.0.1, 2008-10-10, initial
# version 0.1.0, 2008-10-16, similarity and cluster parameters
# version 0.2.0, 2008-10-17, got rid of hankel
# version 0.3.0, 2008-10-17, added Toeplitz SSA
# version 0.3.1, 2008-10-19, C code for crossproduct
# version 0.3.2, 2008-10-19, preprocessing option
# version 0.3.3, 2008-10-19, C code for fromCross
# version 0.4.0, 2008-10-19, added Fourier SSA
#
# to do:
#
# -- add clustering methods
# -- add similarity measures
# -- add preprocessors
# -- ndim for eigen
# -- better algorithm for Toeplitz eigenvectors
#

dyn.load("ssa.so")

ssa<-function(data,L=floor(length(data)/2),b=as.list(1:L),preproc=center,
        ndim=width,method="hankel",ssa_similarity=ssa_w_cor,ssa_cluster=ssa_trans,
            par=.25) {
    if (!identical(class(data),"ts")) stop("Data should be of class ts")
    data<-preproc(data); T<-length(data); K<-T-L+1
```

```
    if (identical(method,"toeplitz")) {
        cross<-toeplitz(rconv(data,1:L))
        s<-eigen(cross)$vectors
        }
    if (identical(method,"hankel")) {
        cross<-hankel(data,L)
        s<-eigen(cross)$vectors
        }
    if (identical(method,"fourier")) {
        s<-sincos(L)
        }
    a<-fromCross(data,s)
    r<-ssa_similarity(a,L,K)
    b<-ssa_cluster(r,par)
    a<-sapply(b,function(kk) rowSums(as.matrix(a[,kk])))
    r<-ssa_similarity(a,L,K)
    y<-ts(a,start=start(data),end=end(data),frequency=frequency(data))
    return(list(y=y,b=b,s=colSums(a^2),r=r))
}

ssa_w_cor<-function(z,l,k) {
    ls<-min(l,k); ks<-max(l,k)
    n<-nrow(z); w<-rep(ls,n)
    w[1:ls]<-1:ls; w[(ks+1):n]<-n-(ks:(n-1))
    c<-crossprod(z,w*z); d<-diag(c)
    return(c/sqrt(outer(d,d)))
}

ssa_trans<-function(r,cut) {
    s<-ifelse(abs(r)>cut,1,0)
    v<-warshall(s)
    h<-unique(v)
    return(apply(h,1,function(v) which(v==1)))
}

warshall<-function(a) {
n<-nrow(a)
for (j in 1:n) {
    for(i in 1:n) {
        if (a[i,j]==1) a[i,]<-pmax(a[i,],a[j,])
        }
    }
return(a)
}

sincos<-function(L) {
    x<-2*pi*(1:L)/L
    M<-floor(L/2)
    if (is.even(L)) N<-M-1 else N<-M
    s<-matrix(0,L,N+M+1)
    for (k in 1:N)
        s[,k]<-sin(x*k)
    for (k in 0:M)
```

```
        s[,N+k+1]<-cos(x*k)
return(apply(s,2,function(z) z/sqrt(sum(z^2))))
}

center<-function(x) x-mean(x)

ident<-function(x) x

rconv<-function(x,lag) {
.C("cconv",
    as.integer(lag),
    as.integer(length(lag)),
    as.double(x),
    as.integer(length(x)),
    as.double(vector("double",length(lag))))[[5]]
}

hankel<-function(x,L){
cc<-rep(0,L*L); T<-length(x)
cl<-.C("hankelC",
        as.double(x),
        as.double(cc),
        as.integer(L),
        as.integer(T))
return(matrix(cl[[2]],L,L))
}

fromCross<-function(x,s) {
    L<-nrow(s); T<-length(x)
    aa<-rep(0,T*L); s<-as.vector(s)
    al<-.C("fromCrossC",
            as.double(x),
            as.double(s),
            as.double(aa),
            as.integer(L),
            as.integer(T))
    return(matrix(al[[3]],T,L))
}

is.even<-function(x) return((as.integer(x) %% 2) == 0)
```

A.2. **C Code.**

```

void
cconv (int *l, int *m, double *x, int *n, double *s)
{
double *y, *z, *u;
int i;
#pragma omp parallel for default(shared) private(i) schedule(dynamic)
for (i = 0; i < *m; i++)
    {
```

```
    y = x + (*n - l[i]);
    z = x + l[i];
    u = x;
    s[i] = 0.0;
    while (u < y)
    s[i] += *u++ * *z++;
    }
}

void
hankelC(double *x, double *c, int *l, int *t)
{
int i, j, h, K, L, T; double s;
L = *l; T=*t; K = T - L + 1;
for (j = 1; j <= L; j++)
        for (h = j; h <= L; h++) {
                s = 0.0;
                for (i = 1; i <= K; i++)
                        s += x[i+j-2] * x[i+h-2];
                c[j + (h - 1) * L - 1] = c[h + (j - 1) * L - 1] = s;
                }
}

void
fromCrossC(double *x, double *s, double *a, int *l, int *t)
{
    int i, j, k, h, ilw, iup, ni, K, L, T;
    double sv, sw;
    L = *l; T=*t; K = T - L + 1;
    for (i = 1; i <= T; i++) {
        ilw = (i + 1) - L; if (ilw < 1) ilw = 1;
        iup = i; if (iup > K) iup = K;
        ni = iup - ilw + 1;
        for (j = 1; j <= L; j++) {
            sv = 0.0;
            for (k = ilw; k <= iup; k++) {
                sw = 0.0;
                for (h = 1; h <= L; h++)
                    sw += x[k + h - 2] * s[h + (j - 1) * L - 1];
                sv += sw * s[(i + 1) - k + (j - 1) * L - 1];
            }
        a[i + (j - 1) * T - 1] = sv / ((double) ni);
        }
    }
}
```

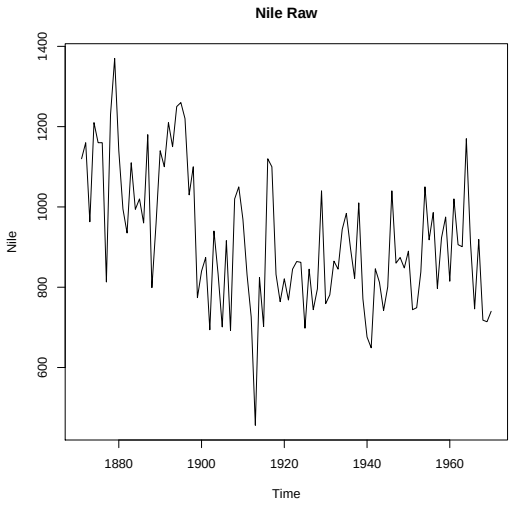
Department of Statistics, University of California, Los Angeles, CA 90095-1554

E-mail address, Jan de Leeuw: deleeuw@stat.ucla.edu

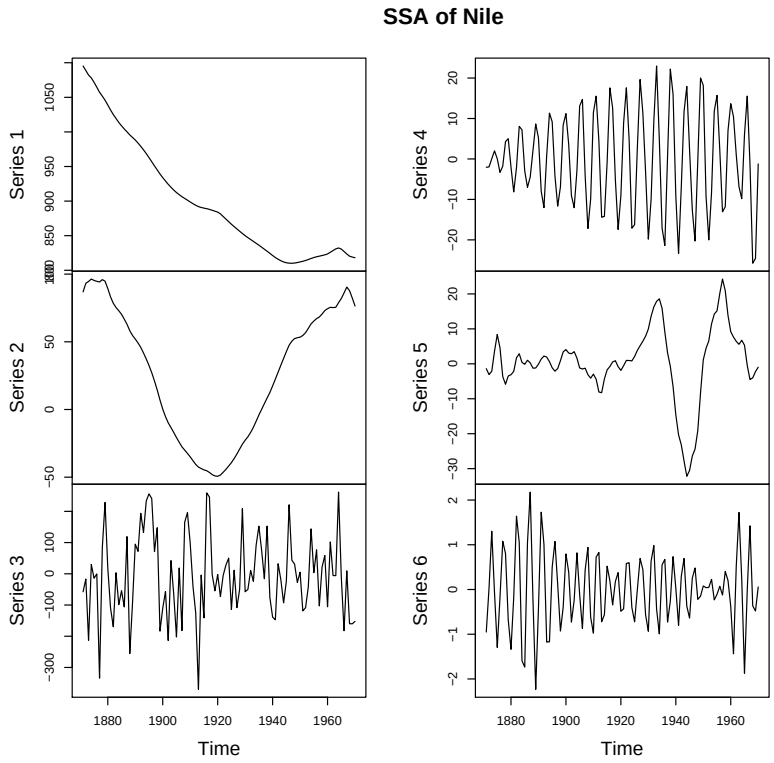
URL, Jan de Leeuw: http://gifi.stat.ucla.edu

E-mail address, Patrick Crutcher: pcrutcher@stat.ucla.edu

URL, Patrick Crutcher: http://www.stat.ucla.edu/~pcrutcher

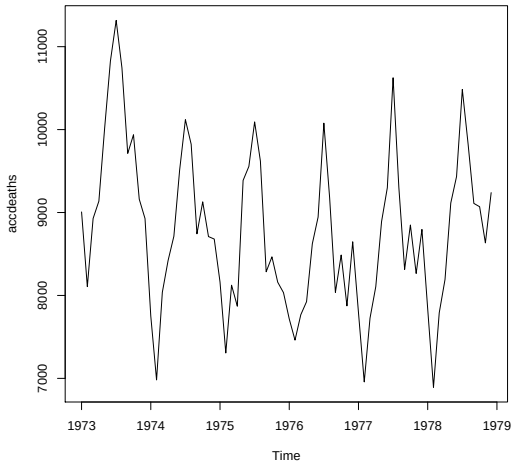


(a) Nile Data

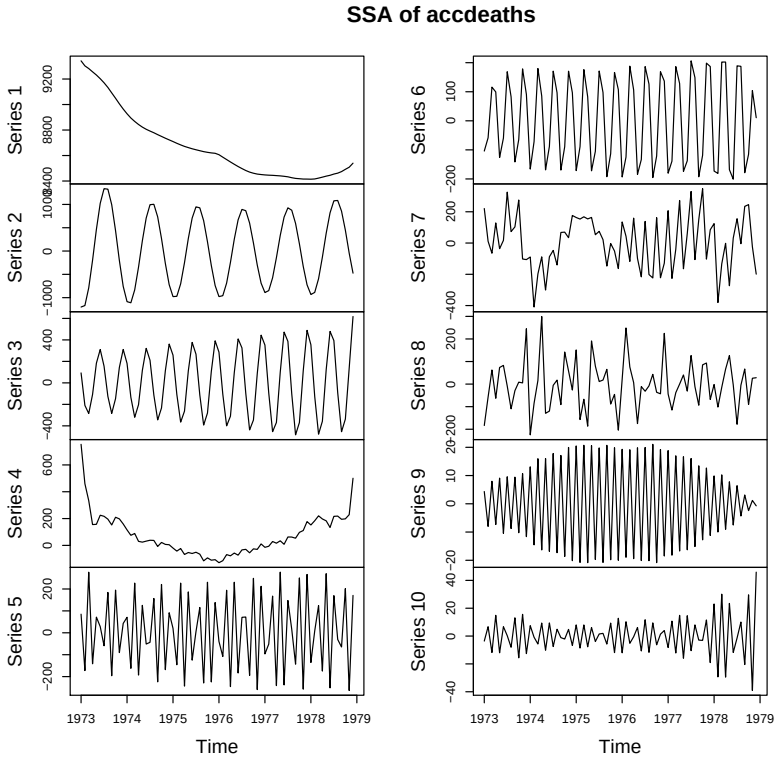


(b) Nile Data, window width 50, Cut-off 0.25

FIGURE 1. Nile Data

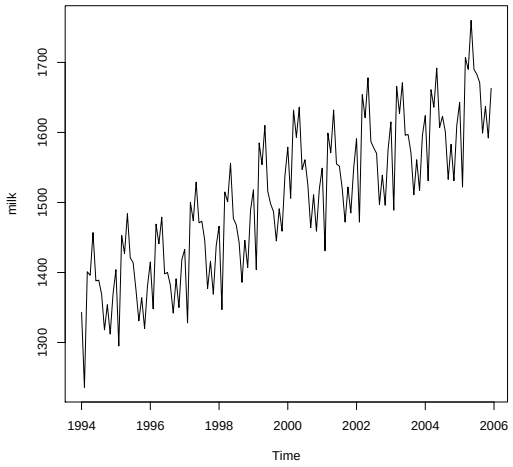


(a) Accidental Deaths Data

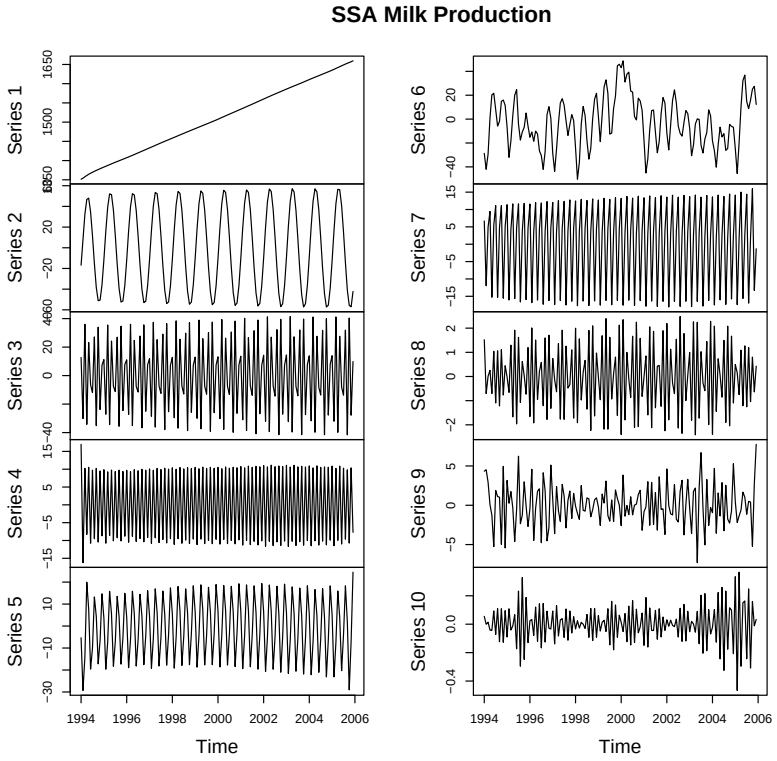


(b) Accidental Deaths Data, window width 36, Cut-off 0.25

FIGURE 2. Accidental Deaths

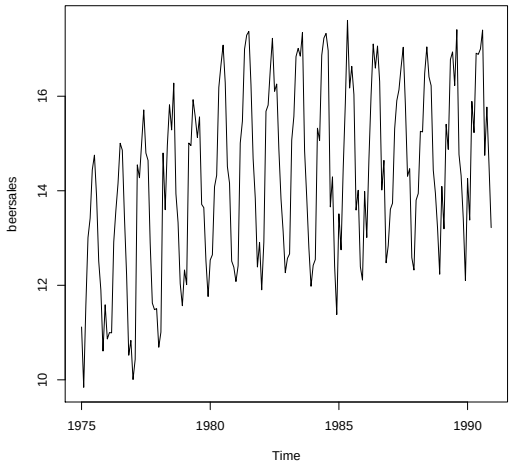


(a) Milk Yield Data

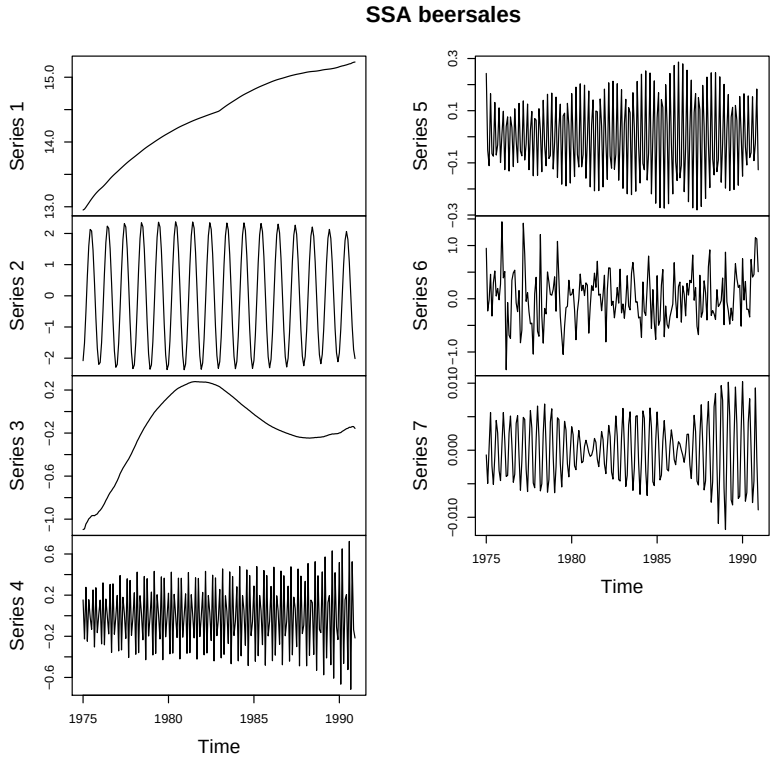


(b) Milk Yield Data, window width 72, Cut-off 0.25

FIGURE 3. Milk Yields



(a) Beer Sales Data



(b) Beer Sales SSA, window width 96, Cut-off 0.25

FIGURE 4. Beer Sales

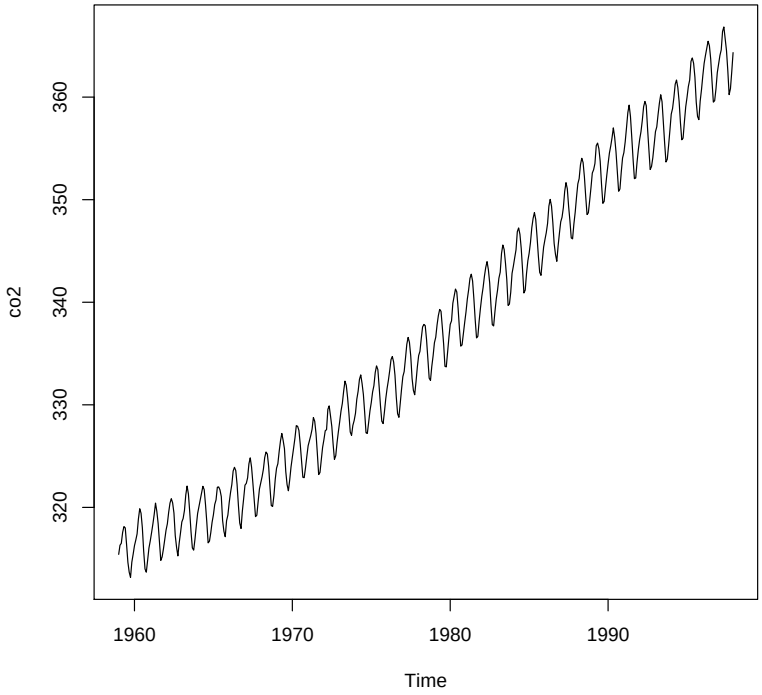
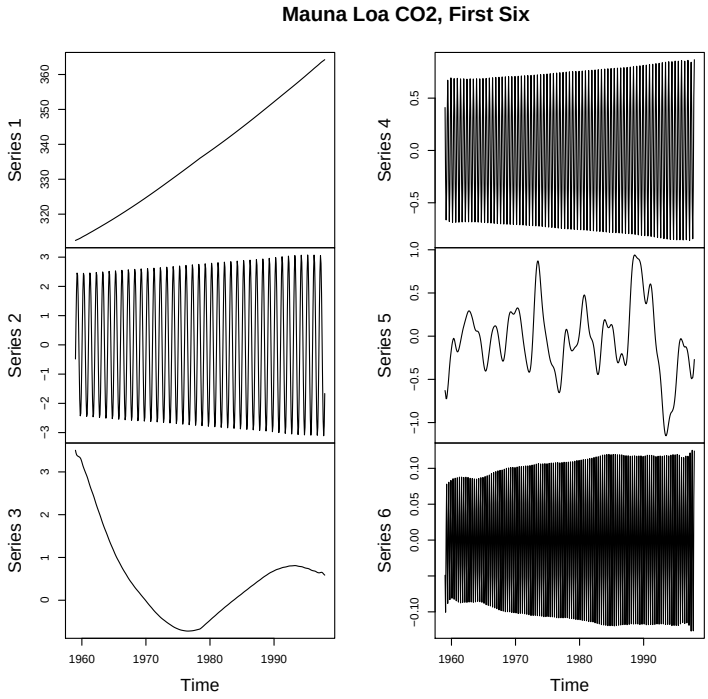
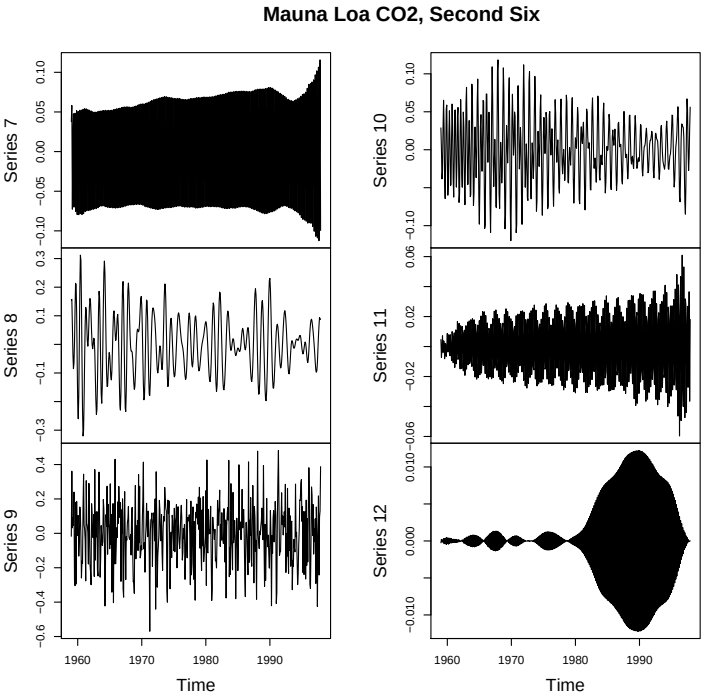


FIGURE 5. Mauna Loa CO_2 Data

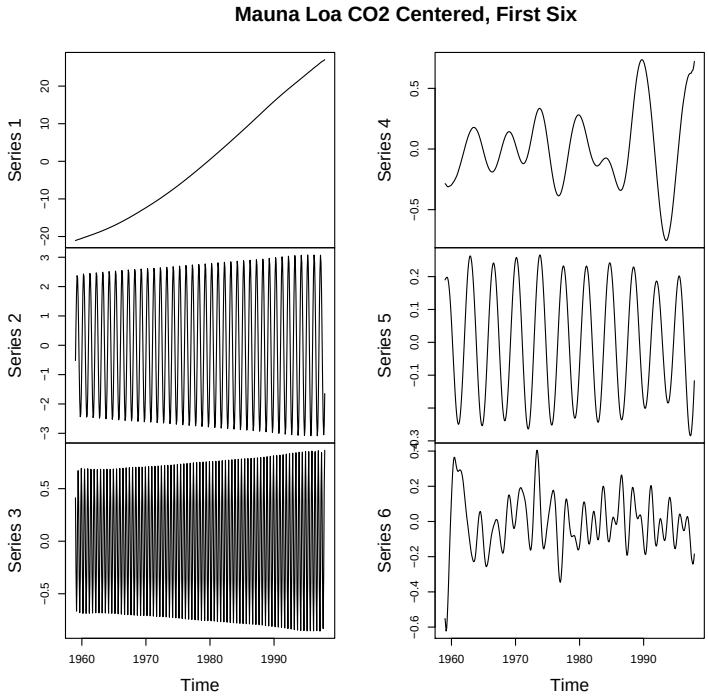


(a) Mauna Loa CO_2, $L = 234$, $\epsilon = 0.25$

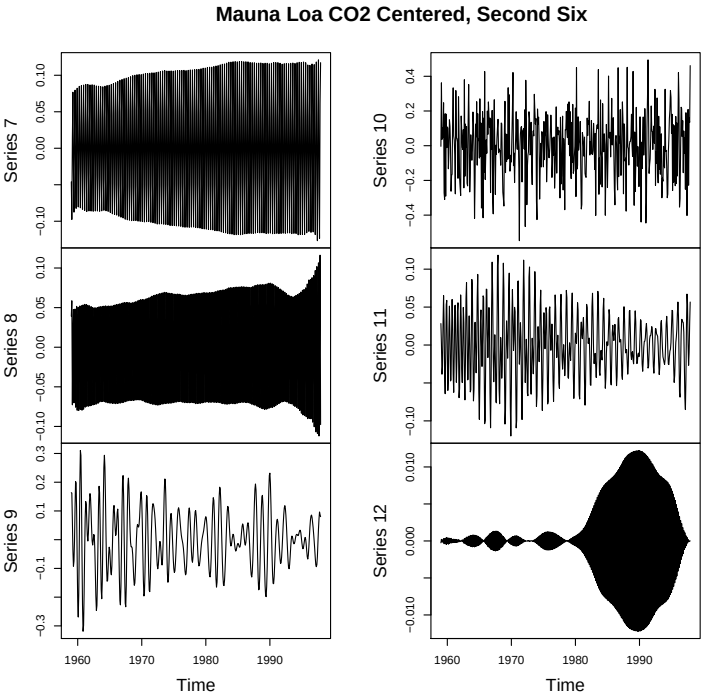


(b) Mauna Loa CO_2, $L = 234$, $\epsilon = 0.25$

FIGURE 6. Mauna Loa CO_2



(a) Mauna Loa CO_2 Centered, $L = 234$, $\epsilon = 0.25$



(b) Mauna Loa CO_2 Centered, $L = 234$, $\epsilon = 0.25$

FIGURE 7. Mauna Loa CO_2 Centered

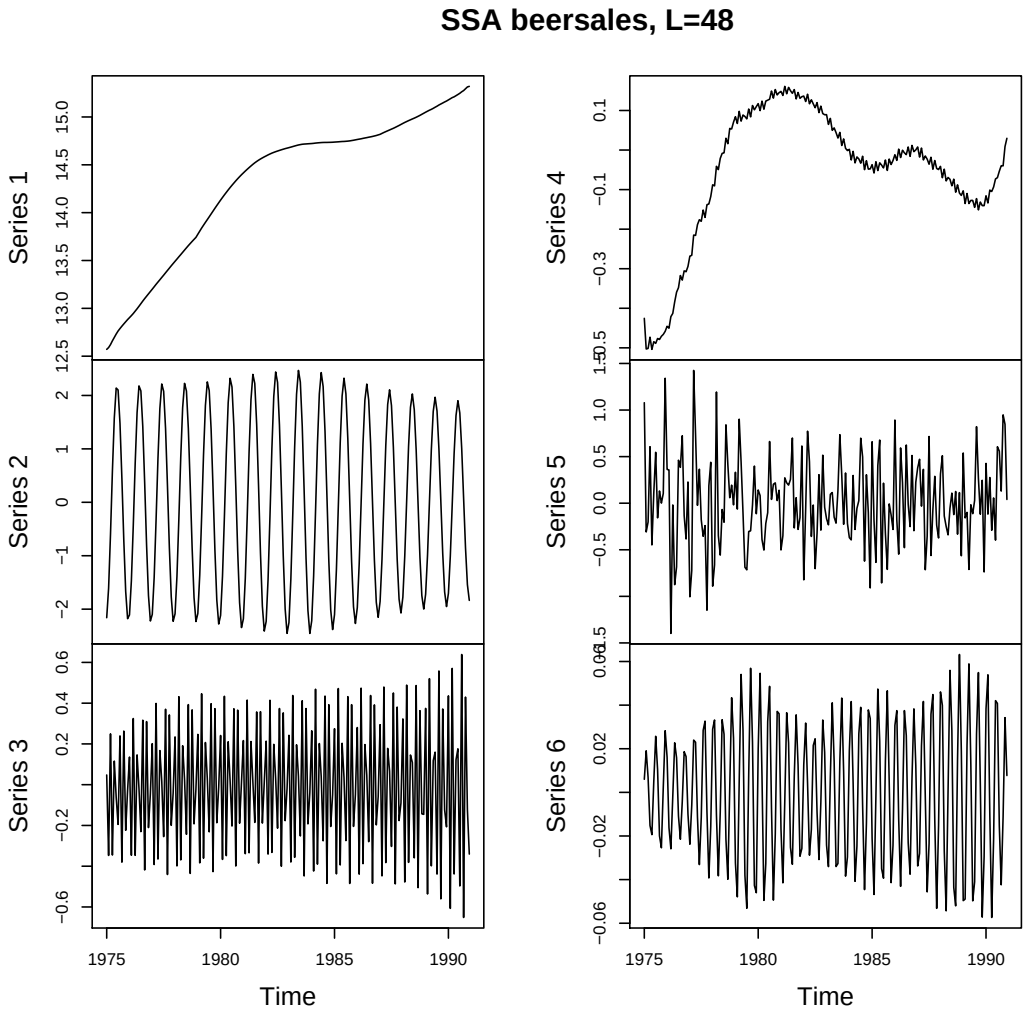
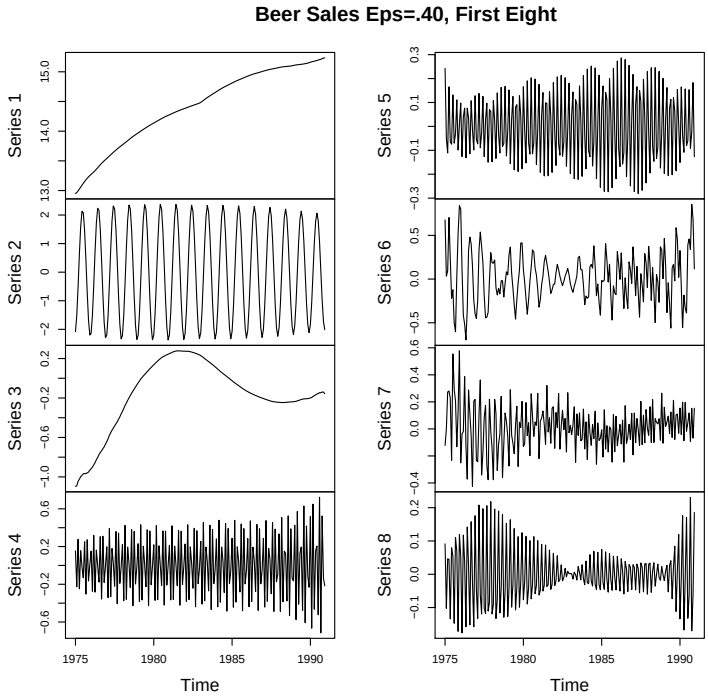
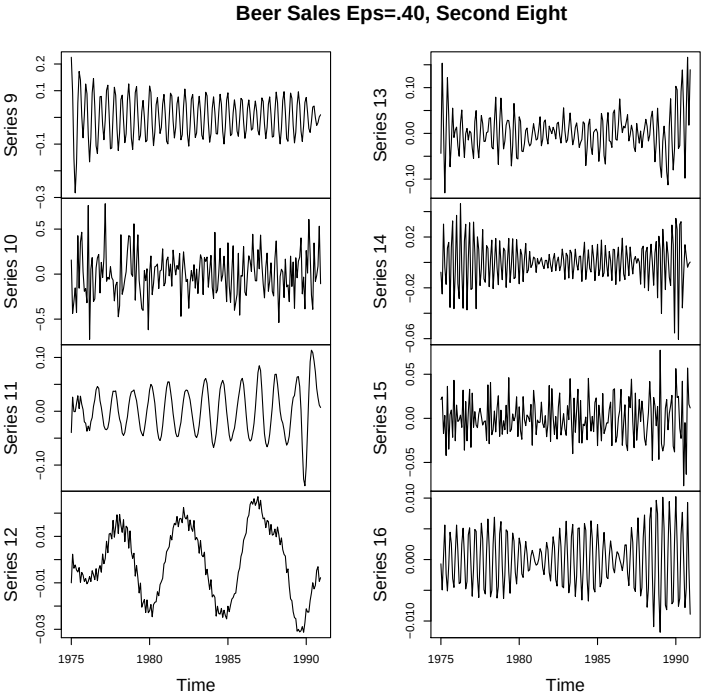


FIGURE 8. Beer Sales, $L = 48$



(a) Beer Sales $\epsilon = 0.40$, First Eight



(b) Beer Sales $\epsilon = 0.40$, Second Eight

FIGURE 9. Beer Sales $\epsilon = 0.40$