Notes on Single Spectrum Analysis using PCA on Timeseries

Notes on single spectrum analysis (SSA) are drawn from the book by Joliffe “Principal Component Analysis 2ed”. [1]

The SSA is an application of PCA on a lagged timeseries for a single variable X the lags are defined as

$$x_t, x_{t+1}, \cdots, x_{t+p-1}$$

where p represents the window of the time lag. A matrix is constructed of the time lags, and the principle component analysis is performed on the matrix of dimension $n-p+1$ rows and p columns. The resulting eigenvectors after the principle component decomposition are trigonometrics functions, also known as “Empricial Orthogonal Functions” EOFs.

Jolliffe indicates the following properties for the derived principle components and eigenvalues.

1. Principle Components are moving averages of the time series.
2. The Eigenvectors (EOFs) provide the weights of the moving averages.
3. The SSA for a timeseries with an oscillatory component, has an associated pair of EOFs with identifical eigenvalues. Coefficients of the EOFs (the eigenvectors) have the same oscillatory pattern but are $\frac{\pi}{2}$ out of phase with each other. [1]

The application of SSA is to determine the dominant periodicities in a series. However, statistical significance tests need to determine whether the periods are due to signal or noise (commonly referred to as red noise). Such inference processes depende on a monte carlo test.

The following example makes use of sensor readings for precipitation made available by the Australian Bureau of Meterology available at http://www.bom.gov.au/climate/change/datasets/datasets.shtml.

For this example the site “CAPE MORETON LIGHTHOUSE” sensor “040043” was selected to obtain the univariate timeseries readings of daily rainfall (millimeters) for the date range starting at 01-01-1888 until 09-02-2018. The data was preprocessed prior to generating the principle components (for an example of reading all sensor data for all stations refer to the “debug.R” script in the git repository for this notebook).

The initial step is to load the preprocessed time series.

```r
series <- read.csv("data/example_cape_moreton.csv", stringsAsFactors = FALSE)
series$date <-  as.POSIXct(strptime(series$date,"%Y-%m-%d"))
idx <- !is.nan(series$precip)
## Need to start the series from January 1888 to start "at the "zero" the time series.
series <- series[series$date >= "1888-01-01",]

nullids <- which(is.na(series))

series <- series[!is.nan(series$precip),]
series <- series[!is.na(series$precip),]

str(series)
```

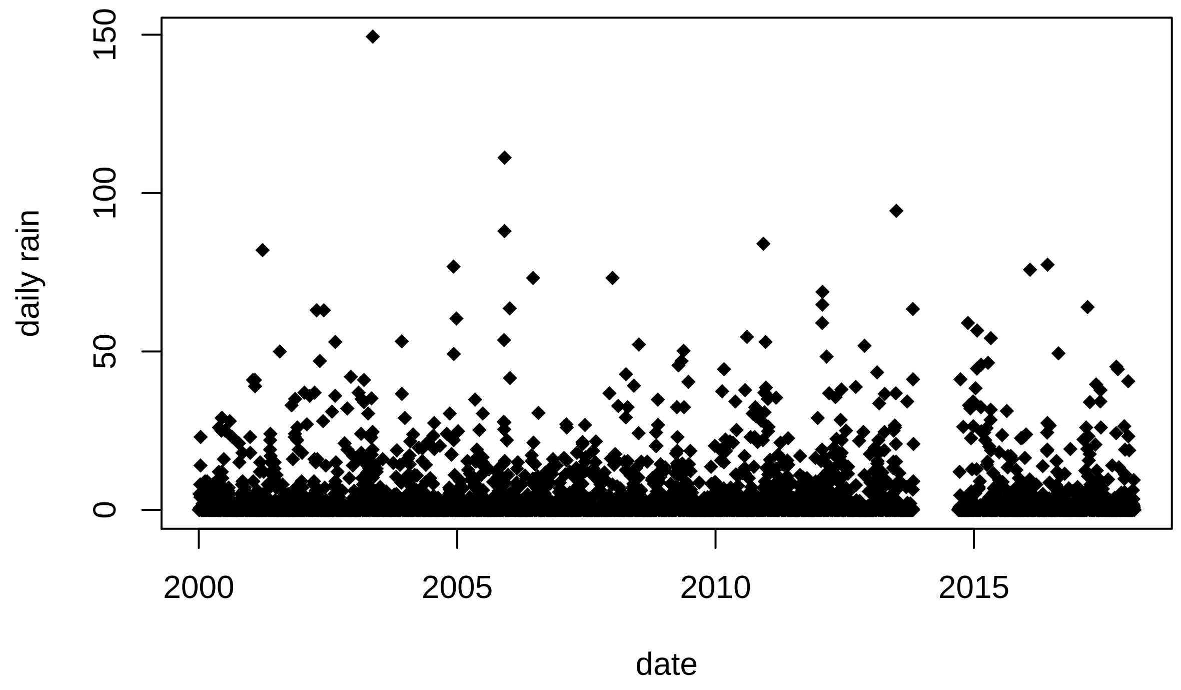
```
## 'data.frame':    46757 obs. of  2 variables:
##  $ date  : POSIXct, format: "1888-01-01" "1888-01-02" ...
##  $ precip: num  2 0 0 0 1.5 0 0 0 0 0 ...
```

```
summary(series)
```

```
##       date                         precip       
##  Min.   :1888-01-01 00:00:00   Min.   :  0.000  
##  1st Qu.:1920-03-02 00:00:00   1st Qu.:  0.000  
##  Median :1952-03-03 00:00:00   Median :  0.000  
##  Mean   :1952-04-17 19:06:01   Mean   :  4.033  
##  3rd Qu.:1984-03-04 00:00:00   3rd Qu.:  2.300  
##  Max.   :2018-02-09 00:00:00   Max.   :339.900
```

Reviewing a subset of the series, say from 2000 onwards, to get an idea of the data.

```
subset <- series[series$date > "2000-01-01",]
plot(subset$precip ~ subset$date, ylab="daily rain", xlab="date", pch=18)
```



Note there was also a period where the sensor must have been offline, somewhere prior to 2015. The missing sensor data would need to be somehow accounted for in the signal, in this example, there has not been any treatment of the missing sensor data, however it may be necessary to perform multiple separate analysis over several regions of the time series where sensor data is available. Over the series there does appear to be a set of peaks, although in this plot these are not clear.

The next step is to lag the data over a period of 365 days, note for the sake of this example the missing data has been dealt discard. But no further checks for missing data has been carried out, so in a more serious analysis, this does require attention, as the sequence of samples needs to be consistent over the time range).

```
sequence <- series$precip[!is.nan(series$precip)]
M <- slidingMatrix(sequence, 365)
dim(M)
```

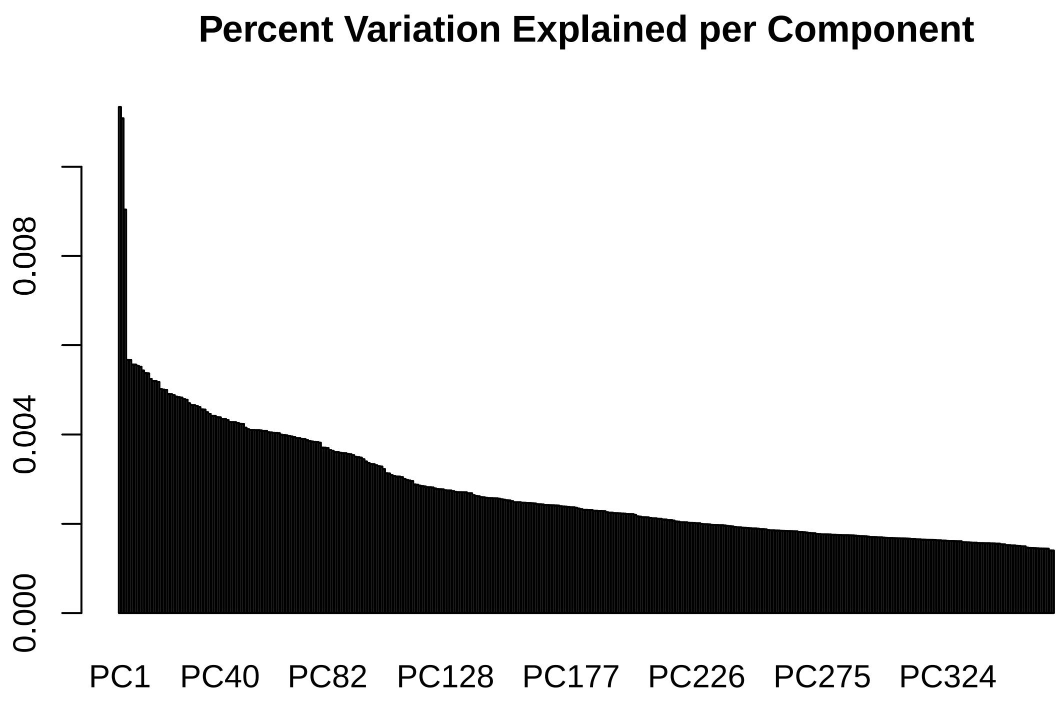
```
## [1] 46393   365
```

The principle component analysis is performed on the lagged matrix.

```
M.pc <- princomp(M, cor=TRUE)
```

The percentage of variation explained for each principle component is plotted.

```
## We seek the percentage of variance to determine the amount of variation explained
## by each respective component within the time series.
percentVar <- M.pc$sdev^2/(sum(M.pc$sdev^2))

barplot(percentVar, names.arg=paste("PC", 1:length(percentVar), sep=""), main="Percent Variation Explai
```



```
df <- data.frame(comp=1:length(percentVar), percent=percentVar, cumpercent=cumsum(percentVar))
```

The first 10 components are shown below.

```
df[1:10,]
```

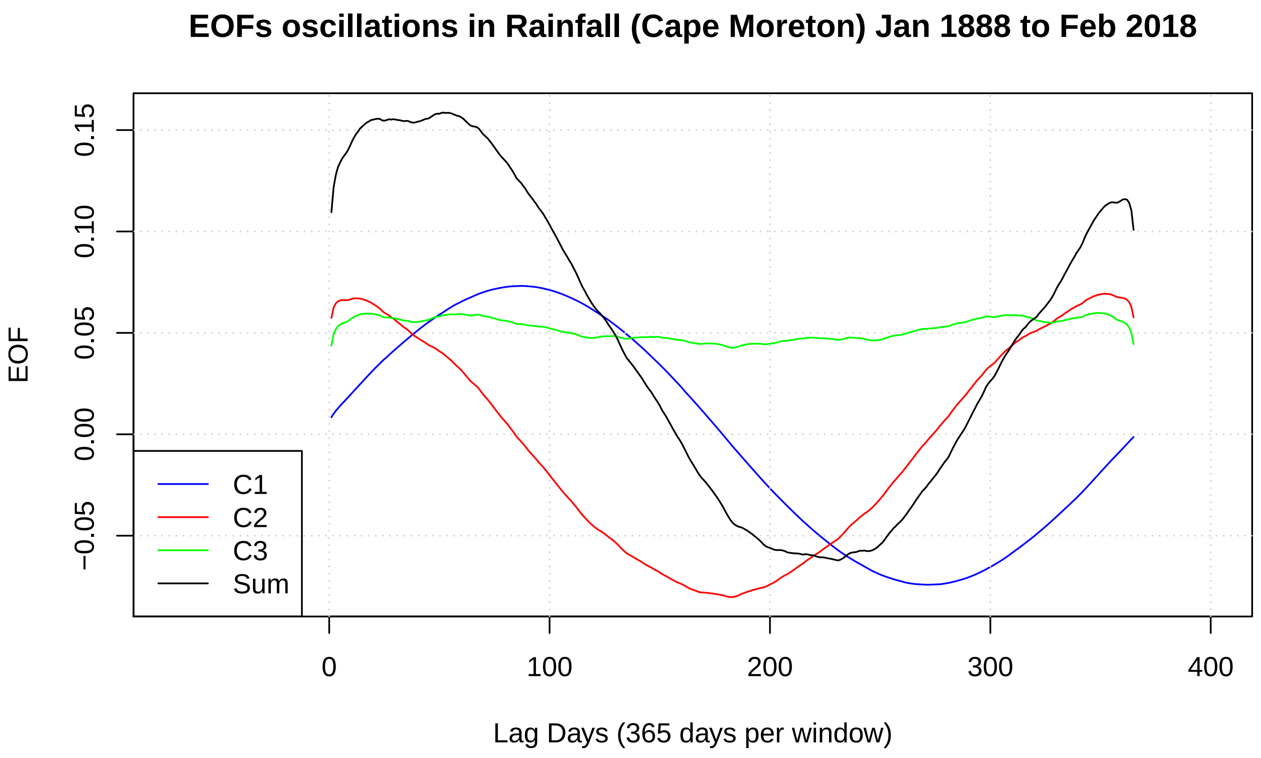
```
##         comp     percent cumpercent
## Comp.1     1 0.011338515 0.01133851
## Comp.2     2 0.011086259 0.02242477
## Comp.3     3 0.009041899 0.03146667
## Comp.4     4 0.005676192 0.03714287
## Comp.5     5 0.005671334 0.04281420
## Comp.6     6 0.005568883 0.04838308
## Comp.7     7 0.005568227 0.05395131
## Comp.8     8 0.005542621 0.05949393
## Comp.9     9 0.005520183 0.06501411
## Comp.10   10 0.005435212 0.07044932
```

Note that the percent of accumulated variance explained by the first three components is only 3 percent. This is quite low, perhaps it is expected given the high amount of variability in the data set.

The eigenvectors associated with the highest three principle components are selected, as well as their combination and plotted over the timerange of the window to show the oscillations of the top three components.

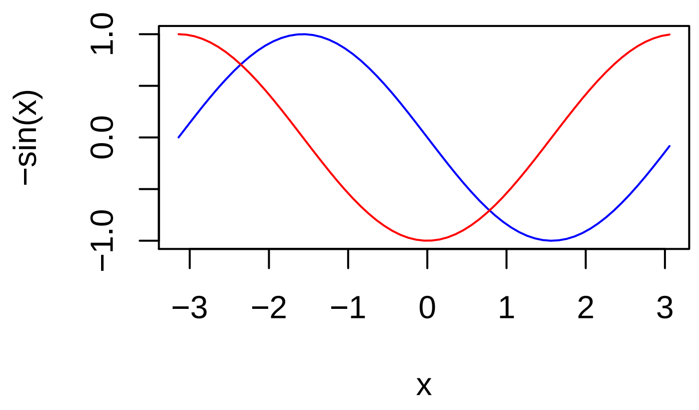
```
## The eigen vectors provide the periods of oscillation associated with the
ev1 <- M.pc$loadings[,1]
ev2 <- M.pc$loadings[,2]
ev3 <- M.pc$loadings[,3]
```

```
sumEv <- ev1 + ev2 + ev3

ylimits <- c(min(ev1,ev2,ev3, sumEv), max(ev1,ev2,ev3, sumEv))

plot(ev1, type="l", col="blue",
     ylim=ylimits,
     xlim=c(-70,400),
     xlab="Lag Days (365 days per window)",
     ylab="EOF",
     main=paste("EOFs oscillations in Rainfall (Cape Moreton)", strftime(min(series$date), "%b %Y"), "t
grid(col = "lightgray", lty = "dotted",
     lwd = par("lwd"), equilogs = TRUE)
lines(ev2, col="red")
lines(ev3, col="green")
lines(sumEv)
legend("bottomleft", lty=c(1,1,1), col=c("blue", "red", "green", "black"), legend=c("C1","C2","C3", "Su
```



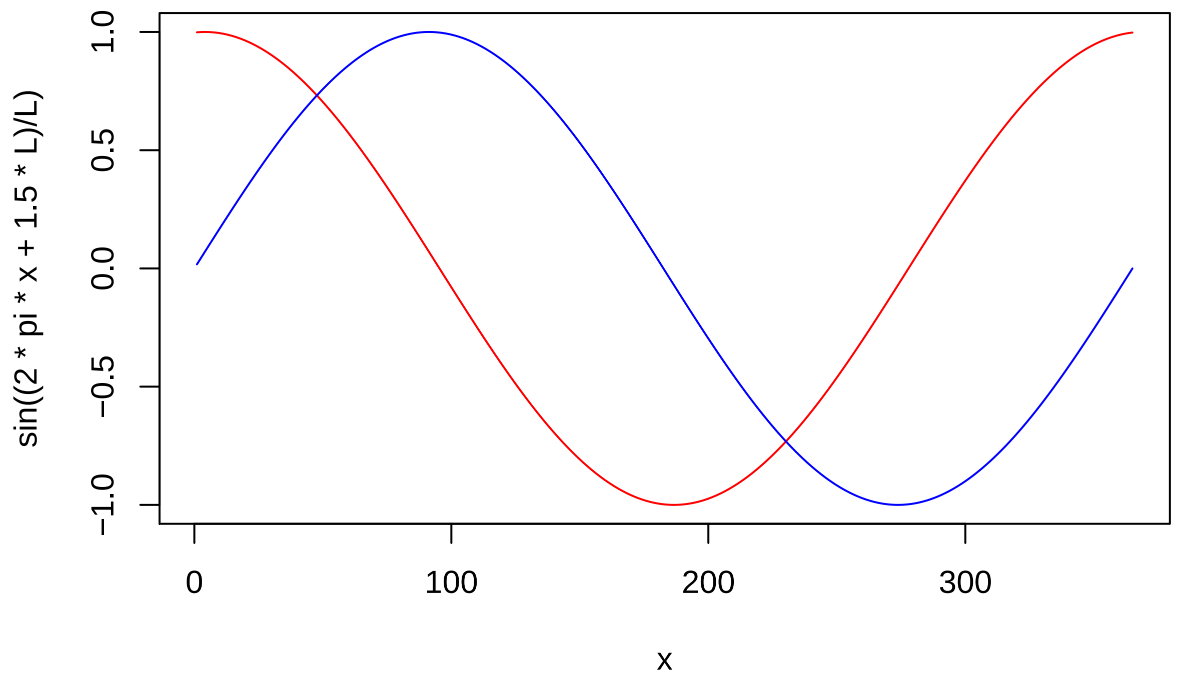
Note the first two components indeed refect trigonometric functions both slightly out of phase with each other. This is illustrated below with two example overlapping sin functions, $-sin(x)$ and $-sin(x + \frac{\pi}{2})$ over the range $[-\pi, \pi]$.

```
x <- seq(from=-pi, to=pi, by=0.1)
plot(x, -sin(x), col="blue", type="l")
lines(x, -sin(x+pi/2), col="red")
```



We can simulate the period of $L = 365$.The offset is fudged below to match the above diagram.

```
x <- seq(from=1, to=365, by=0.1)
L <- 365
plot(x, sin((2*pi*x + 1.5*L)/L), col="red", type="l", ylim=c(-1.0,1.0))
lines(x, sin((2*pi*x)/L), col="blue")
```



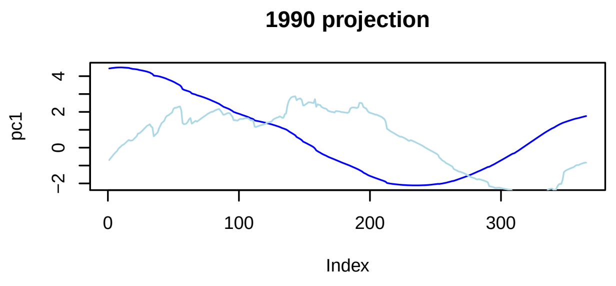
The first two eigenvectors as indicated by the first two eigenvalues in the PCA represent the most influential oscillations in the data set. The period of the oscillation (between higher and lower levels of rain) being approximately 200 days. Aside from this, the seasonality of the oscillation loosely reflects summary in the early and latter part of the year which does tend to have alot more rainfall in the region than winter.

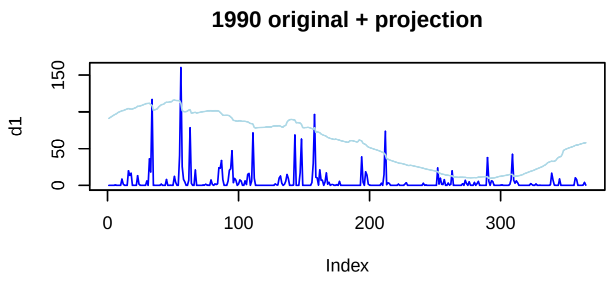
The plot below displays the first two projections of principle components for the years 1990, 1991, 1992 and 1993. The variability of the signal overlaps between the first two projections, although it is difficult to determine whether this variation cycles with any periodicity.

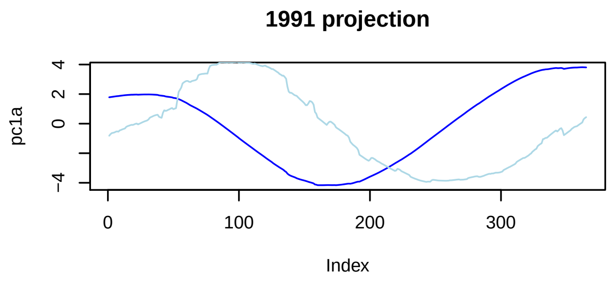
```
subset <- series$date[!is.nan(series$precip)]
idx1 <- which(subset >= "1990-01-01")
idx2 <- which(subset <= "1990-12-31")
idx <- intersect(idx1, idx2)[1:365]
```

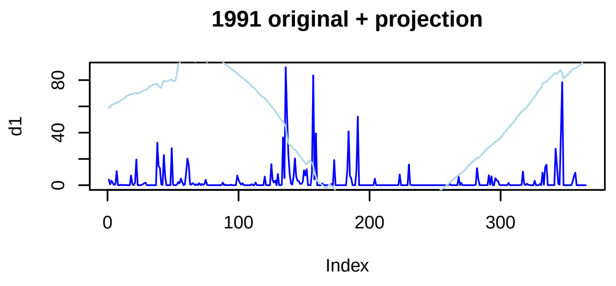
```r
## Projection of original data onto the first two components given by the scores.
pc1 <- M.pc$scores[idx,1]
s1 <-  M.pc$scores[idx,2]
delta1 <- pc1-s1
par.old <- par(mfrow=c(4,2))
#xlimits <- c(min(series[idx,]$date), max(series[idx,]$date))
plot(pc1, col="blue", type="l", main="1990 projection")
lines(s1, col="lightblue")

## original data
d1 <- sequence[idx]
d2 <- M.pc$scale*(pc1 + s1 + M.pc$center)
plot(d1, col="blue", type="l", main="1990 original + projection")
lines(d2, col="lightblue")

idx1 <- which(subset >= "1991-01-01")
idx2 <- which(subset <= "1991-12-31")
idx <- intersect(idx1, idx2)[1:365]
pc1a <- M.pc$scores[idx,1]
s1 <-   M.pc$scores[idx,2]
delta2 <- pc1a-s1
plot(pc1a, col="blue", type="l", main="1991 projection")
lines(s1, col="lightblue")


## original data
d1 <- sequence[idx]
d2 <- M.pc$scale*(pc1a + s1 + M.pc$center)
plot(d1, col="blue", type="l",  main="1991 original + projection")
lines(d2, col="lightblue")

idx1 <- which(subset >= "1992-01-01")
idx2 <- which(subset <= "1992-12-31")
idx <- intersect(idx1, idx2)[1:365]
pc1a <- M.pc$scores[idx,1]
s1 <-  M.pc$scores[idx,2]
delta3 <- pc1a-s1

plot(pc1a, col="blue", type="l", main="1992 projection")
lines(s1, col="lightblue")


## original data
d1 <- sequence[idx]
d2 <- M.pc$scale*(pc1a + s1 + M.pc$center)
plot(d1, col="blue", type="l",  main="1992 original + projection")
lines(d2, col="lightblue")


idx1 <- which(subset >= "1993-01-01")
idx2 <- which(subset <= "1993-12-31")
idx <- intersect(idx1, idx2)[1:365]
pc1a <-M.pc$scores[idx,1]
```

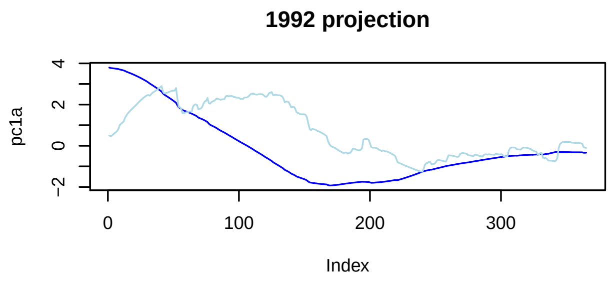
```
s1 <-  M.pc$scores[idx,2]
delta4 <- pc1a-s1

plot(pc1a, col="blue", type="l", main="1993 projection")
lines(s1, col="lightblue")


## original data
d1 <- sequence[idx]
d2 <- M.pc$scale*(pc1a + s1 + M.pc$center)
plot(d1, col="blue", type="l",  main="1993 original + projection")
lines(d2, col="lightblue")
```

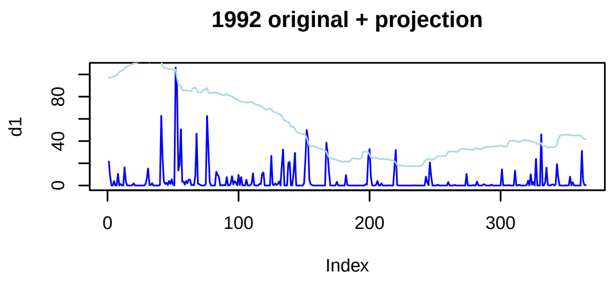


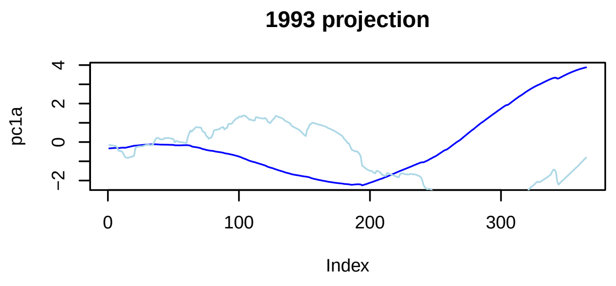


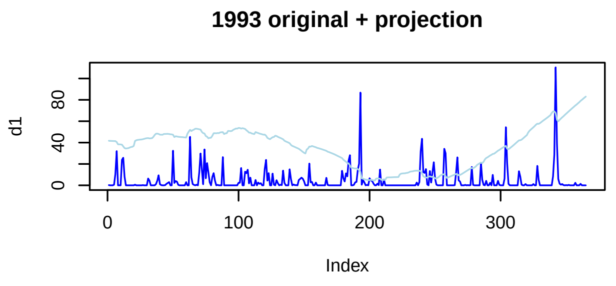












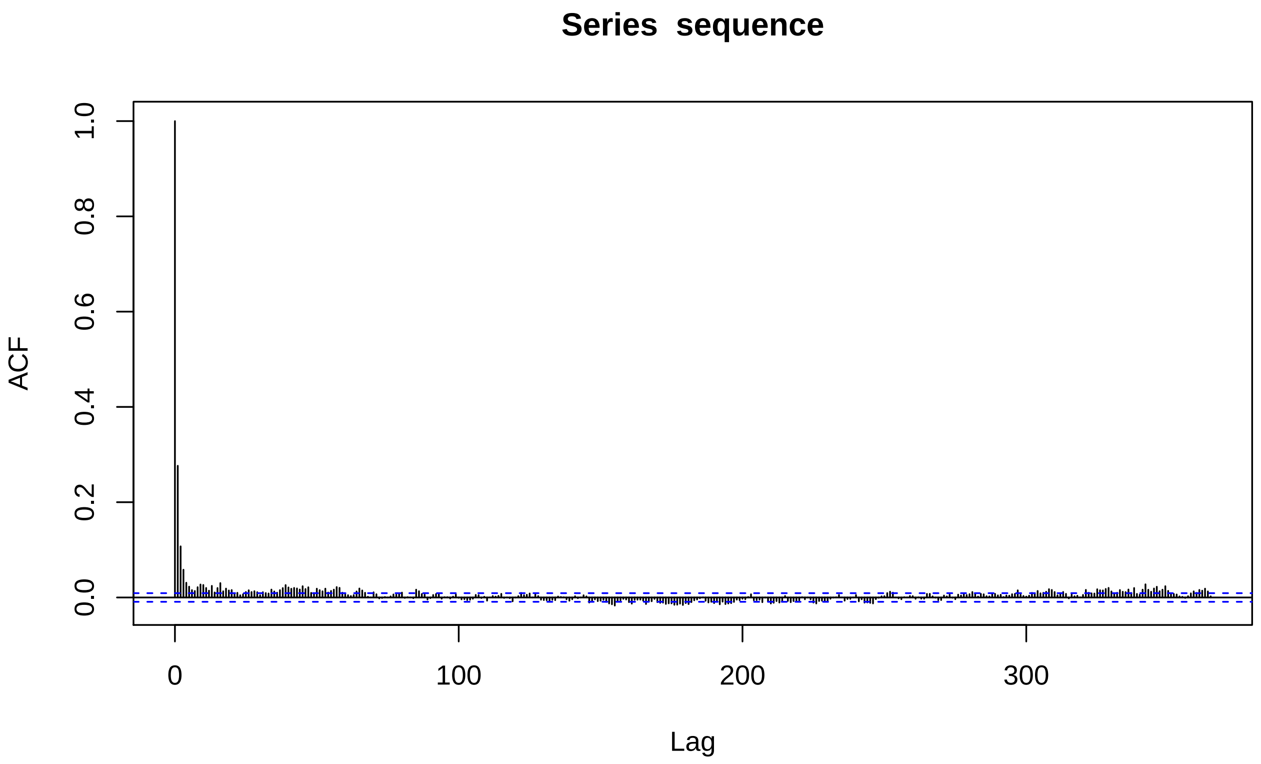
```r
par(par.old)
```

The question is whether these oscillations are due to signal within the data, or whether it is simply a result of noise.

Allen and Smith [2] propose a statistical test against the null hypothesis that the resulting decomposition is of the same distribution as that generated by the combination of the original signals correlation and the correlation given by red noise (generated by a purely random component). A method of monte carlo simulation is used to generate the red noise component. The procedure is known as *Monte Carlo singular spectrum analysis*.

The autocorrelation measure for the series is given as:

```r
autoC <- acf(sequence, lag.max=365)
```



[1] Jolliffe I. T. (2002). *Principle Component Analysis*, Second Edition. Springer Series in Statistics, New York

[2] Allen, M R and Smith, L A (1996). Monte carlo ssa: Detecting irregular oscillations in the presence of colored noise. *Journal of Climate.* **9** 3373–404