Afrika Statistika

Afrika Statistika
Vol. 13 (1), 2018, pages 1499–1509.
DOI: http://dx.doi.org/10.16929/as/1499.116

ISSN 2316-090X

Spatio-temporal Predictions using Multivariate Singular Spectrum Analysis

Richard Awichi[1*],

Busitema University, P.O Box 236, Tororo, Uganda

Selected Papers presented at the East African Conference of Statistical Mathematics and Applications, EACSMA 2017

Received September 28, 2017; Accepted February 18, 2018

Abstract. . In this paper, we present a method for utilizing the usually intrinsic spatial information in spatial data sets to improve the quality of temporal predictions within the framework of singular spectrum analysis (SSA) techniques. The SSA-based techniques constitute a model free approach to time series analysis and ordinarily, SSA can be applied to any time series with a notable structure. Indeed it has a wide area of application including social sciences, medical sciences, finance, environmental sciences, mathematics, dynamical systems and economics. SSA has two broad aims: i) To make a decomposition of the original series into a sum of a small number of independent and interpretable components such as a slowly varying trend, oscillatory components and a structure-less noise. ii) To reconstruct the decomposed series for further analysis in the absence of the noise component. Multivariate singular spectrum analysis (MSSA) is an extension of SSA to multivariate statistics and takes advantage of the delay procedure to obtain a similar formulation as SSA though with larger matrices for multivariate data. In situations where spatial data is an important focus of investigation, it is not uncommon to have attributes whose values change with space and time and an accurate prediction is thus important. The usual question asked is whether the intrinsic location parameters in spatial data can improve data analysis of such data sets. The proposed method is based on the Inverse Distance Weighting and is exemplified on climate data. Results show that the proposed technique of incorporating spatial dependence into MSSA analysis leads to improved quality of statistical inference.

Key words: Time Series Analysis; MSSA; Inverse Distance Weighting; Spatial Dependence.
AMS 2010 Mathematics Subject Classification : 62H11; 62M15

*Corresponding author Richard Awichi : ichbinrao@gmail.com

Résumé (French) Dans cet article, nous présentons une méthode qui utilise les informations spatiales intrinsèques de données spatiales pour améliorer la qualité des prédictions temporelles dans le cadre de techniques d'analyse multivariée spectrale singulière (MSSA). La question habituelle posée est savoir si les paramètres de localisation intrinsèques des données spatiales peuvent améliorer l'analyse de telles données. La méthode proposée est basée sur la notion Inverse Distance Weighting. Elle est appliquées à des données climatiques. Les résultats montrent que la technique proposée d'intégration de la dépendance spatiale dans l'analyse MSSA conduit à une amélioration de la qualité de l'inférence statistique.

1. Introduction

Singular Spectrum Analysis (SSA), a well developed tool for time series analysis, is a model free approach to time series analysis, as opposed to model based time series analysis with several restrictive assumptions. SSA has two broad aims: i) to make a decomposition of the original series into a sum of a small number of independent and interpretable components such as a slowly varying trend, oscillatory components and a structureless noise and ii) to reconstruct the decomposed series for further analysis in the absence of the noise component. SSA is implemented through a sequence of steps and the following are the steps in brief, detailed exposition and background theory can be found in Golyandina *et al.* (2001). The first step is the *Embedding* step in which the time series $F_N = (f_1, \cdots, f_N)$ is transformed into a multidimensional data matrix $\mathbf{X}$, called the trajectory matrix using an embedding operator: $\mathcal{T}(F) \longrightarrow \mathbf{X}$. The single most important parameter in this step is the window length L. The second step is the *Singular value decomposition* (SVD) step in which the trajectory matrix is factorized into a sum of elementary matrices using the nonzero eigenvalues of $\mathbf{X}\mathbf{X}^T$. The *Grouping* step is the third step where the elementary matrices are split further through the procedure known as *eigentriple grouping*. The final step is the *Diagonal averaging* or also commonly known as *Hankelization*. This step transfers the sum of the elementary matrices after eigentriple grouping back to the time series. It is in a way the reverse of step one. A very important concept in SSA is the notion of Separability, see Golyandina, N. (2010). Multivariate Singular Spectrum Analysis (MSSA) is a direct extension of SSA to multivariate analysis and takes advantage of the (delay) embedding procedure to obtain a similar formulation as SSA, albeit with larger matrices for multidimensional time series. The main aim of MSSA is to extract signal from the multivariate time series leaving out the residual (noise) so as to perform further analysis, see Patterson *et al.* (2011).

2. Inverse Distance Weighting, IDW

For spatial data sets, there is always the intrinsic (geographic) information–the location attribute embedded in every recording. Whether it is in environmental sciences, economics, agriculture, climatology, geology or any other fields where spatial data is frequently encountered, the desire to harness this embedded information for purposes of analysis cannot be over emphasized. Data mining is an automated search for knowledge hidden in large collections of data set attributes. In environmental science and other areas where space-time behaviour is an important focus of investigation, it is not uncommon to have attributes

whose values often change with space and time. This leads to spatial dependence which subsequently influences data analysis, see Müller, W. G. (2007) and therefore a technique to incorporate spatial information into the analysis of such data sets is desirable. Data close together in space and time usually exhibit higher dependencies than those that are farther apart, see Cressie N. A (1993). This dependency is thus inversely proportional to the distance of separation between any two data gathering sites. A method that utilizes this inverse proportionality in the distance of separation as a technique of incorporating spatial dependence into the analysis is the inverse distance weighting (IDW). Inverse distance weighting assigns bigger weights at near points and smaller weights at distant locations. There are several ways of computing the inverse distance weights to be used in the analysis, see Mateu, J. and Müller, W. G. (Eds.). Here, we present the inverse distance technique first introduced in Awichi, R. O. and Müller, W. G. (2013).
The multivariate data set $y = \{y_{ij}\}$ is an $s \times N$ matrix and the inverse distance is given as $w_{ij} = 1/d_{ij}$ where d_{ij} is the Euclidean distance between locations i and j. For missing values in the data, a new weight is calculated by excluding the corresponding distance measure from the w_{ij}s. To include spatial information into the analysis based on the model free MSSA framework, we premultiply the data set by the row-normalized spatial weight matrix, $W = \{w_{ij}\}$ to yield the spatially weighted averages Wy.

3. Data

The proposed technique was applied to climate data from several recording sites in Upper Austria. The data was provided by the Zentralanstalt für Metereologie in Austria and is described in more detail in Mateu, J. and Müller, W. G. (Eds.). This data set contains climatic data measured at 37 stations irregularly placed over the region provided from http://www.zamg.ac.at/fix/klima/oe71-00/klima2000/klimadaten_oesterreich_1971_frame1.htm.
Here, we have (incomplete) monthly data from Jan 1994 to Dec 2009 on average temperature and total rainfall. Due to some missing observations, however, not all of the stations could be effectively used. A map of the region with the respective locations of the measurement stations and the contours for the rainfall data is displayed in Figure 1.

For purposes of this application, we used the rainfall data from 11 locations that have no missing information, hence the series length is 192 (monthly recordings from Jan 1994 to Dec 2009). The rest of the sites have missing data to varying degrees of missingness. The data was also preprocessed by log-transformation, see Golyandina, N. and Zhigljavsky, A. (2013).
To determine the effect of the spatial dependence on the data, we pooled together the data at different levels. The pooling was done by conditioning data from a particular site, y_i on data from the rest of the sites and likewise for the spatially weighted averages. To assess the accuracy, we calculated the root mean square errors, RMSE, i.e. $R(y_i|y)$ and $R(y_i|Wy)$. The RMSE $R(y_i)$ of the single unweighted series is referred to as the default RMSE in this paper. If $R(y_i|y) < R(y_i)$ (or $R(y_i|Wy) < R(y_i)$), then the proposed technique leads to improved quality of the results, otherwise it is worse than results without pooling. For comparative purposes, the ratio of the root mean square error, which is given by $RRMSE = \frac{\sum RMSE}{\sum RMSE(y_i)}$

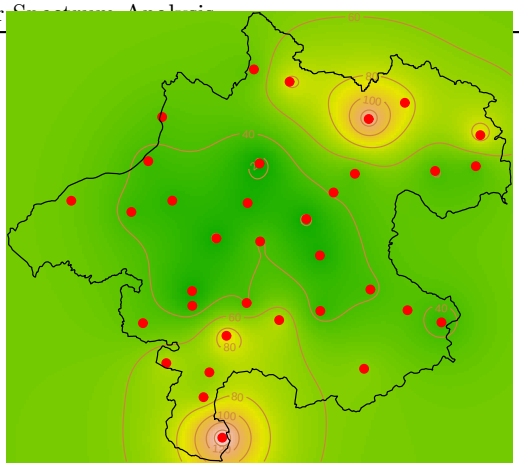

Fig. 1. The Sampling Locations of the Climatic Data Set Within Upper Austria

can be computed. The analysis was done in R using the package `Rssa`, see Korobeynikov *et al.* (2015).

3.1. Rssa Package

.

The most time consuming step of SSA is the SVD and this is overcome in Rssa by using computationally efficient algorithms described in details in Korobeynikov, A. (2010). The fastest implementation of SSA is done in the R package Rssa, and since only a few leading components are required, Rssa uses the so-called Partial SVD to compute the required number of leading eigentriples, see Golyandina, N. and Zhigljavsky, A. (2013). Another advantage is the Hankel nature of the trajectory matrix which helps in speeding up the matrix multiplications for the optimum value of the window length $L \sim N/2$. The entry point to the package is the function `ssa`, which performs the decomposition stage. The function has the following format:

$$ssa(x, L, ..., kind, svd.method, force.decompose = TRUE).$$

Further details of the arguments can be viewed by poring through the 'help' page from the package download. However, for the format indicated above, the argument `x` gives the input series, `L` specifies the window length, `kind` corresponds to the type of SSA analysis to be performed (can be "1d-ssa" or "toeplitz-ssa" or "2d-ssa" or any other kind), `svd.method` allows for the selection of the SVD method to be used and the default value `force.decompose = TRUE` enables completion of the Decomposition stage, see Korobeynikov, A. (2010). The acceleration in the Rssa package is achieved in the following ways:

– combining the embedding step with the SVD step thereby decreasing the storage requirement,

– applying the Lanczos-based Partial SVD for computational efficiency in calculating the eigentriples of the required components, see Korobeynikov, A. (2010),
– applying the Fast Fourier Transform (FFT) in the multiplication of a Hankel matrix by a vector thus reducing the computational complexity of the SVD step and also at the reconstruction stage.

The main step of the SSA method is the singular value decomposition of the series trajectory matrix. The package provides several implementations of this procedure (this corresponds to the different values of `svd.method` argument). Some of these implementations are briefly discussed below, details can be found in Korobeynikov, A. (2010) and the help page of the package from R, see for example R Core Team. (2014).

– `auto`. This is the automatic method of selection depending on the series length,N, the window length L, the SSA kind and number of eigenvalues requested.
– `nutrlan` and `propack`. These methods utilize the Lanczos-based Partial SVD techniques and the Hankel structure of the trajectory matrix for efficient computations and are quite fast. The methods use the truncated SVD (where only a specified number of eigentriples are to be computed) and the continuation of the decomposition. `propack` is slightly faster and more numerically stable, see Korobeynikov, A. (2010) for more information.
– `svd`. The method does not assume anything special about the trajectory matrix and thus is slow.
– `eigen` This method computes full SVD via eigen-decomposition of the cross-product matrix. It is faster than the `svd` method, but still slow for most matrix sizes.

Usually the `ssa` function tries to provide the best SVD implementation for given series length and the window size. In particular, for small series and window sizes it is better to use generic black-box routines (as provided by `svd` and `eigen` methods). For long time series, special purpose routines are to be used. The outcome of the function `ssa` is an *SSA object* which is the input for the majority of other functions in the package. The contents of the object can be viewed via the function `summary`. There are several other functions in the package, for example the function `reconstruct(x, groups)` is used to perform Reconstruction stage, where the first argument is an SSA object and the second specifies the eigentriple grouping. For these and other details about the Rssa package, see Golyandina, N. and Zhigljavsky, A. (2013) and for the latest version of the package, see Korobeynikov *et al.* (2015).

The R codes we used in our computations are partly reported in the Appendix.

4. Results

The results for the in-sample analyses can be found in Awichi, R. O. and Müller, W. G. (2015). Here, we present the out-of-sample analyses. For MSSA, the window length, L can be calculated using the relation $L \leq \frac{sN}{s+1}$, where N = time series length and s = dimension of the series, see Golyandina, N. and Zhigljavsky, A. (2013). A good choice of the window length ensures proper separability. We report findings for the unweighted and the spatially weighted for the set of entire sites. The other set of results fall between these two. For the

out-of-sample analysis, we used the data up to Dec 2008 implying that the window length reduced to 180. The data for the final year, Jan-Dec 2009, was used to compare with the predicted values for the same period for different forecast procedures. To assess the effect of the spatial weighting, we computed both RMSE and the mean absolute percentage deviation, MAPD for the different forecast steps ($MAPD = \frac{1}{M}\sum_{t=1}^{M}\left(\frac{|y_t-\hat{y}_t|}{|y_t|}\right)$).

Table 1. Out-of-Sample Forecasts

Forecast for 2009	Actual Value	Weighted Forecast	Unweighted Forecast	Weighted		Unweighted	
				MAPD	RMSE	MAPD	RMSE
Jan	3.044522	2.939382	5.208310	M=1:			
Feb	4.174387	2.770864	4.487440	0.0345	0.1051	0.7107	2.1638
Mar	4.369448	3.915668	3.894846	M=3:			
Apr	3.091042	3.860014	4.259645				
May	4.615121	4.376068	3.735367	0.1582	0.8538	0.2981	1.2917
Jun	5.347108	3.931153	4.622641	M=6:			
Jul	4.997212	5.077728	3.817844				
Aug	4.317488	4.812715	3.637416	0.1733	0.8982	0.2664	1.1306
Sept	3.931826	4.607082	4.456604	M=12:			
Oct	4.127134	4.629895	3.005752				
Nov	3.637586	3.189317	2.980330				
Dec	3.433987	4.079065	2.969817	0.1480	0.7313	0.2261	0.9876

Table 1 shows the results for one of sites, Freistadt. The spatially weighted forecasting conditioning on Wy outperforms the default forecast at all levels of forecast steps. This performance can be checked against the RMSE and MAPD values for the spatially weighted and the unweighted forecasts.
A rolling forecast was undertaken for the spatially weighted series for the same selected site. Results of the comparison with the year-long forecast is shown in Table 2.

5. Conclusions

Using IDW to incorporate spatial dependence into the analysis of spatial data within the framework of MSSA time series analysis leads to improved quality of statistical analysis. This can be seen from Table 1, where both the MAPD and RMSE values are smaller for spatially weighted analysis where spatial lag (weight) matrix was used to incorporate spatial dependence into the analysis. The year-long prediction outperforms the rolling forecast potentially due to the seasonality within the original time series.
We therefore highly recommend incorporation of spatial dependence (via spatial weight matrix) into the model free time series analysis within the framework of the SSA-based techniques. For prediction, the inherent characteristics or features of the time series under investigation should be taken into consideration before deciding upon the method to use.

Table 2. Comparison of Rolling and Year Long Predictions

Actual y_t	Year-long $\hat{y}_t$	Rolling $\hat{y}_t$	Year-long $\lvert y_t - \hat{y}_t \rvert$	Rolling $\lvert y_t - \hat{y}_t \rvert$
3.044522	2.939382	2.939382	0.105140	0.105140
4.174387	2.770864	2.559900	1.403523	1.614487
4.369448	3.915668	3.380577	0.453780	0.988871
3.091042	3.860014	3.468199	0.768972	0.377157
4.615121	4.376068	3.696763	0.239053	0.918357
5.347108	3.931153	3.308346	1.415954	2.038761
4.997212	5.077728	4.753443	0.080516	0.243769
4.317488	4.812715	4.575633	0.495227	0.258145
3.931826	4.607082	5.613441	0.675257	1.681616
4.127134	4.629895	7.590778	0.502761	3.463644
3.637586	3.189317	6.127708	0.448269	2.490121
3.433987	4.079065	6.221942	0.645078	2.787955
		MAPD	0.1480	0.3518
		RMSE	0.7313	1.7716

References

Awichi, R. O. and Müller, W. G. (2013). Improving SSA Predictions by Inverse Distance Weighting. *Revstat Statistical Journal*, **11(1), 105–119**.

Awichi, R. O. and Müller, W. G. (2015). In-Sample Spatio-temporal Predictions by Multivariate Singular Spectrum Analysis. *Procedia Environmental Sciences* **26, 19 – 23**.

Cressie N. A (1993). *Statistics for Spatial Data*. Revised Edition, Wiley.

Golyandina, N., Nekrutkin, V., and Zhigljavsky, A. (2001). *Analysis of Time Series Structure: SSA and Related Techniques*. Chapman & Hall; CRC, New York - London.

Golyandina, N. and Zhigljavsky, A. (2013). *Singular Spectrum Analysis for Time Series*. Springer.

Golyandina, N. and Anton Korobeynikov (2014). Basic Singular Spectrum Analysis and forecasting with R. *Computational Statistics and Data Analysis*. **71, 934–954**.

Korobeynikov, A. (2010). Computation and space efficient implementation of SSA. *Statistics and Its Interface*; **Vol.3, 357–368**.

Golyandina, N., Korobeynikov, A., Shlemov, A. and Usevich, K. (2013). Multivariate and 2D Extensions of SSA with Rssa Package. *arXiv:1309.5050V1[stat.ML]*.

Golyandina, N. (2010). On the Choice of Parameters in Singular Spectrum Analysis and related space-based methods. *Statistics and Its Interface*; **Vol.3, 259–279**.

Korobeynikov, A., Shlemov, A., Usevich, K. and Golyandina, N. (2015). Rssa: A collection of methods for singular spectrum analysis http://CRAN.R-project.org/package=Rssa. R package version 0.13.

Mateu, J. and Müller, W. G. (Eds.) (2012). *Spatio-temporal design: Advances in efficient data acquisition*; (Statistics in Practice). Wiley.

Müller, W. G. (2007). *Collecting Spatial Data*. Third Edition, Springer.

R Core Team. (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Patterson, K., Hassani, H., Heravi, S. and Zhigljavsky, A. (2011). Multivariate Singular Spectrum Analysis for Forecasting Revisions to Real-time Data. *Journal of Applied Statistics*, **38:10, 2183- 2211.**

APPENDIX.

R Codes for Spatially Weighted Analysis

These R codes are to calculate the spatial weight matrix to incorporate the spatial information into the analysis.

```
RainData4 <- read.csv("UpperaFULLDATA.csv", sep = ",")
xm <- as.matrix(RainData4[, 2:3])
d1 <- dist(xm)
distancematrix <- as.matrix(d1)
dt1 <- 1/distancematrix
dt1[dt1==Inf] <- 0
dist <- dt1
beta <- apply(dist, 1, sum)
dist1 <- sweep(dist, 1, beta, "/")
Rains <- RainData4[,-c(1:3)]
Rains <- as.matrix(Rains)
LogRains <- log(Rains)
LogRains[LogRains==-Inf] <- 0
LogRains1 <- LogRains
Rains2 <- dist1%*%Rains
LogRains2 <- log(Rains2)
Rdata <- as.matrix(cbind(RainData4[, c(1:3)], Rains2))
write.csv(t(LogRains1), file = "LogRains1.csv")
write.csv(t(LogRains2), file = "LogRains2.csv")
library("Rssa")
LogRains14 <- read.table("LogRains14.txt", header = TRUE)
LogRains14 <- ts(LogRains14[,-1], start = c(1994, 1),
                 end = c(2009, 12), frequency = 12)
Rain14 <- window(LogRains14)
St14=Rain14[,"St1"] #Vary "St1" till "St11"
s.St14 <- ssa(St14, L = 96, kind = "1d-ssa")
r.St14 <- reconstruct(s.St14, groups = list(Seasonality = 2:11))
r.St142 <- reconstruct(s.St14, groups = list(Signal = 1:42))
p.St14 <- Reduce("+", r.St142)
plot(r.St142, add.residuals = TRUE, add.original = TRUE,
     plot.method = "xyplot",
    superpose =TRUE, auto.key = list(columns = 2))
plot(s.St14, type = "vectors", idx = 1:12)
plot(s.St14, type = "paired", idx = 2:11, plot.contrib = FALSE)
parestimate(s.St14, groups = list(2:3,4:5), method = "esprit-ls")
plot(wcor(s.St14, groups = 1:42), scales = list(at = c(10,20,30,40)))
plot(reconstruct(s.St14, add.residuals = FALSE, add.original = FALSE,
    groups = list(G12 = 2:3, G4 = 4:5, G6 = 6:7, G2.4 = 8:9)))
f.St14 <- vforecast(s.St14, groups = list(Signal = 1:42),
```

```
   len = 12, only.new = TRUE)
plot(cbind(St14, f.St14), plot.type = "single", col = c("black",
     "red"), ylab = NULL)
wt14 <- Rain14[,c("St1","Wt1","Wt2","Wt3","Wt4","Wt5","Wt6","Wt7",
     "Wt8","Wt9","Wt10","Wt11")]
L <- 180 #Different L values used (96,144,156,168,180)
s.wt14 <- ssa(wt14, L = L, kind = "mssa")
r.wt14 <- reconstruct(s.wt14, groups = list(Trend = c(1,6),
     Seasonality = c(2:5, 7:12)))
p.wt14 <- Reduce("+", r.wt14)
r.wt142 <- reconstruct(s.wt14, groups = list (Signal = 1:42))
Rwt14 <- r.wt142$Signal
#plot(r.wt142, add.residuals = FALSE, plot.method = "xyplot",
    superpose = TRUE, auto.key = list(columns = 3))
plot(s.wt14, type = "vectors", idx = 1:12)
plot(s.wt14, type = "paired", idx = 2:14, plot.contrib = FALSE)
parestimate(s.wt14, groups = list(2:3, 4:5), method = "esprit-ls")
plot(wcor(s.wt14, groups = 1:42), scales = list(at = c(10,20,30,40)))
f.wt14 <- rforecast(s.wt14, groups = list(Signal = 1:42) ,len = 12,
    only.new =  TRUE)
plot(cbind(wt14[,"St1"], f.wt14[,"St1"]), plot.type = "single",
    col=c("black", "red"), ylab = "wt14")
rme1 <- sqrt(mean((r.St142$Signal - Rain14[,"St1"])^2))
rme2 <- sqrt(mean((Rwt14[,"St1"]-Rain14[,"St1"])^2))
rme <- c(rme1, rme2)
# Signal Comparison:
wt15 <- Rain14[,c("St1", "Wt1", "St2", "Wt2", "St3", "Wt3")]
s.wt15 <- ssa(wt15, L = 180, kind = "mssa")
r.wt15 <- reconstruct(s.wt15, groups = list(Trend = c(1, 2, 5),
    Seasonality = c(3:4, 6:12)))
plot(r.wt15, add.residuals = FALSE,plot.method = "xyplot",
    slice = list(component = 1), screens = list(colnames(wt15)),
    col = c("blue", "green", "red", "violet", "black", "green4"),
    lty = rep(c(1, 2), each = 6), scales = list(y = list(draw = FALSE)
    ), layout = c(1, 6))
plot(r.wt15, plot.method = "xyplot", add.original = FALSE,
    add.residuals = FALSE, slice = list(component = 2),
    col = c("blue","green","red", "violet", "black", "green4"),
    scales = list(y = list(draw = FALSE)), layout = c(1, 6))
```

R Codes for Computing Rolling Forecasts

These codes are for calculating rolling forecasts from January 2009 to Dec 2009.

```
RainDat6 <- read.csv("UpperaFULLDATARoll.csv", sep = ",")
xm <- as.matrix(RainDat6[, 2:3])
d1 <- dist(xm)
```

```
distancematrix <- as.matrix(d1)
dt1 <- 1/distancematrix
dt1[dt1==Inf] <- 0
dist <- dt1
beta <- apply(dist, 1, sum)
dist1 <- sweep(dist, 1, beta, "/")
RainF16 <- RainDat6[,-c(1:3)]
RainF16 <- as.matrix(RainF16)
LogRainF16 <- log(RainF16)
LogRainF16[LogRainF16==-Inf] <- 0
LogRainF6 <- LogRainF16
RainF26 <- dist1%*%RainF16
LogRainF26 <- log(RainF26)
library("Rssa")
LogRainF16 <- read.table("LogRainF16.txt", header = TRUE)
LogRainF16New <- ts(LogRainF16[,-1], start = c(2009, 1),
                    end = c(2009, 12),frequency = 12)
LogRainF16 <- ts(LogRainF16[,-1], start = c(1994, 1),
                  end = c(2008, 12),frequency = 12)
f.Vwt16 <- vector("numeric", 12)
for (i in 1:12) {
RainF16 <- window(LogRainF16)
V16=RainF16[,"V1"]
s.V16 <- ssa(V16, L = 84, kind = "1d-ssa")
r.V16 <- reconstruct(s.V16, groups = list(Seasonality = 2:11))
r.V162 <- reconstruct(s.V16, groups = list(Signal = 1:42))
p.V16 <- Reduce("+", r.V162)
Vwt16 <- RainF16[,c("V1", "Wt1", "Wt2", "Wt3", "Wt4", "Wt5",
   "Wt6", "Wt7", "Wt8", "Wt9", "Wt10", "Wt11")]
L <- 168
s.Vwt16 <- ssa(Vwt16, L = L, kind = "mssa")
r.Vwt16 <- reconstruct(s.Vwt16, groups = list(Trend = 1,
   Seasonality = c(2:12)))
p.Vwt16 <- Reduce("+", r.Vwt16)
r.Vwt162  <- reconstruct(s.Vwt16, groups = list (Signal = 1:42))
RVwt16 <- r.Vwt162$Signal
f.Vwt16[i] <- rforecast(s.Vwt16, groups = list(Signal = 1:42),
   len = 1, only.new = TRUE)[1]
LogRainF16New[i, "V1"] <- f.Vwt16[i]
LogRainF16 <- ts(rbind(LogRainF16, LogRainF16New[i,]),
                 start = start(LogRainF16),
                 frequency = frequency(LogRainF16))
}
write.csv(f.Vwt16, file = "ForeRollWtd.csv")
```