seasadj simu

linyiguo

2019.5.26

Well, I am here again. In this log, I am going to reproduce the three seasonal adjustment methods from StatCan stuff based on the data I generated in last log **simulate data**.  
The first two should be relatively easier, at least this is what Aaron told me last week…well, I hope so. The first one is X12-ARIMA, which is actually a software package developed by U.S. Census Bureau. In R, we can use package *x12* to access the X12-ARIMA method.

为了能在周一见面的时候给Aaron一份相对满意的答卷，我决定今晚上熬夜奋战一下。毕竟按我现在的结果，是明显不能让他满意的。也的确需要push一下自己，赶一赶进度了。晃晃悠悠地看书，不知道什么时候才能做出来东西了。加油吧，小伙子！今晚上把这两个model的curve给做出来！你可以的！Come On!

*–Update 5.28–* 呵呵呵，Aaron把我鸽了，这也正好给了我多一天的时间来想怎么build the model of state space model.

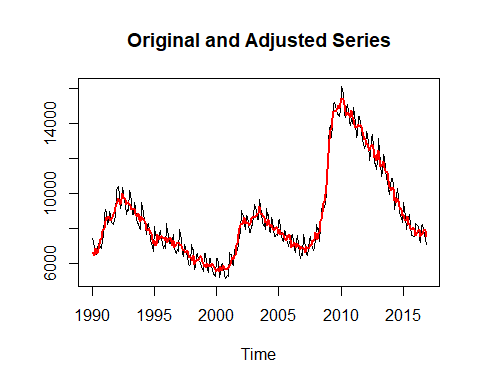
# X-12-ARIMA

学新知识的道路上注定要经历各种弯路，就像我现在一样。According to [wiki](https://en.wikipedia.org/wiki/X-12-ARIMA), X-12-ARIMA is the successor of X-11-ARIMA, and the current version is X-13ARIMA-SEATS(or X-13, and seats means Signal Extraction in ARIMA Time Series). As for the relations between these models, check [website](https://www.r-bloggers.com/seasonal-adjusment-on-the-fly-with-x-13arima-seats-seasonal-and-ggplot2/) and introduction of the [paper](https://cran.r-project.org/web/packages/seasonal/vignettes/seas.pdf). I guess I am going to use the latest version, that is X-13, to remove seasonal component in our ts.  
为了节省时间，也为了降低难度，第一个例子我用上面提到的paper中的数据来进行分析：

library("seasonal")  
# The seas function provides the core functionality of the package, and the 1st argument is 'ts'  
# The unemp example series measures US unemployment and is included in seasonal  
m <- seas(unemp)  
summary(m)

##   
## Call:  
## seas(x = unemp)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## AR-Nonseasonal-01 0.94360 0.03441 27.43 <2e-16 \*\*\*  
## MA-Nonseasonal-01 0.82540 0.05654 14.60 <2e-16 \*\*\*  
## MA-Seasonal-12 0.85071 0.03362 25.30 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## SEATS adj. ARIMA: (1 1 1)(0 1 1) Obs.: 323 Transform: none  
## AICc: 4324, BIC: 4339 QS (no seasonality in final): 0   
## Box-Ljung (no autocorr.): 22.04 Shapiro (normality): 0.9946

plot(m)



# The result is close but not identical to the official seasonally adjusted series.

By default, a call to seas also invokes the following automatic procedures of X-13: \* Transformation selection (log / no log);  
\* Detection of trading day and Easter effects;  
\* Outlier detection;  
\* ARIMA model search.  
In the example above, X-13 opts to perform no pre-transformation, does not adjust for weekday or Easter effects, detects no outliers and models the series by a (1 1 1)(0 1 1) seasonal ARIMA (autoregressive integrated moving average) model. This can be seen from the summary output: There are no coefficients for Easter or outliers, and the ARIMA specification and the transformation are shown at the bottom  
By default, seas calls the *SEATS* adjustment procedure(which decomposes the ARIMA model).To perform the alternative *X-11* adjustment procedure, the following option can be used:

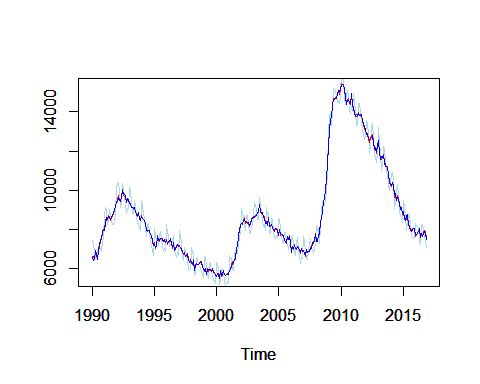
library("seasonal")  
library("seasonalview")

## Registered S3 method overwritten by 'xts':  
## method from  
## as.zoo.xts zoo

##   
## Attaching package: 'seasonalview'

## The following object is masked from 'package:seasonal':  
##   
## view

eg\_seats <- seas(unemp)  
eg\_x11 <- seas(unemp, x11 = "")  
plot(unemp,col="lightblue",ylim=c(5500,15300),ylab="")  
par(new=T)  
plot(final(eg\_seats),col="red",ylim=c(5500,15300),ylab="")  
par(new=T)  
plot(final(eg\_x11),col="blue",ylim=c(5500,15300),ylab="")



seas(x = unemp,arima.model = "(1 1 1)(0 1 1)",regression.aictest = NULL,outlier = NULL,transform.function = "none")

##   
## Call:  
## seas(x = unemp, transform.function = "none", regression.aictest = NULL,   
## outlier = NULL, arima.model = "(1 1 1)(0 1 1)")  
##   
## Coefficients:  
## AR-Nonseasonal-01 MA-Nonseasonal-01 MA-Seasonal-12   
## 0.9436 0.8254 0.8507

Well, the results here seem to be good.

### Let’s try our own data

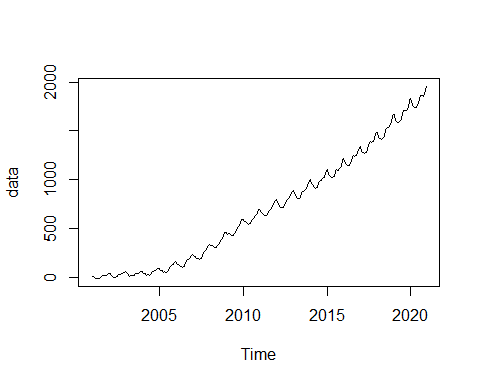
library(forecast)

## Registered S3 methods overwritten by 'ggplot2':  
## method from   
## [.quosures rlang  
## c.quosures rlang  
## print.quosures rlang

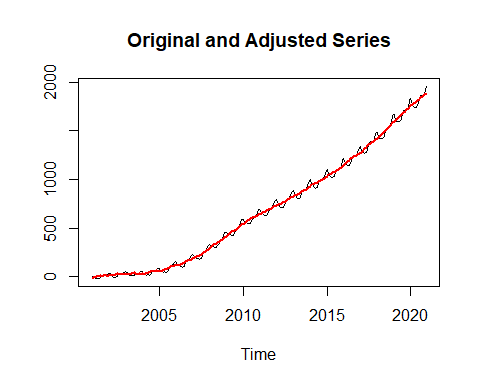
## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':  
## method from   
## fitted.fracdiff fracdiff  
## residuals.fracdiff fracdiff

library(seasonal)  
set.seed(1)  
model <- Arima(ts(rnorm(24000),freq=12), order=c(0,1,1), seasonal=c(0,1,1),fixed=c(theta=0.5, Theta=0.5))  
data <- simulate(model,nsim=240)  
plot(data)



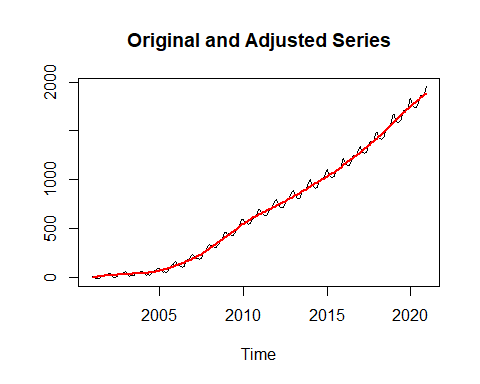
m\_x11 <- seas(data, x11 = "", regression.aictest = NULL)  
plot(m\_x11)



m\_seats <- seas(data, regression.aictest = NULL)

## Model used in SEATS is different: (0 1 1)(0 1 0)

plot(m\_seats)



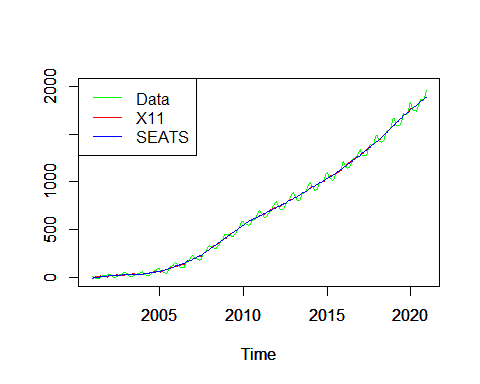
summary(m\_seats)

##   
## Call:  
## seas(x = data, regression.aictest = NULL)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## MA-Nonseasonal-01 -0.47429 0.05798 -8.180 2.85e-16 \*\*\*  
## MA-Seasonal-12 -0.53983 0.05679 -9.506 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## SEATS adj. ARIMA: (0 1 1)(0 1 1) Obs.: 240 Transform: none  
## AICc: 1279, BIC: 1289 QS (no seasonality in final): 0   
## Box-Ljung (no autocorr.): 21.33 Shapiro (normality): 0.9955   
## Messages generated by X-13:  
## Warnings:  
## - Automatic transformation selection cannot be done on a  
## series with zero or negative values.  
##   
## Notes:  
## - Model used for SEATS decomposition is different from the  
## model estimated in the regARIMA modeling module of  
## X-13ARIMA-SEATS.

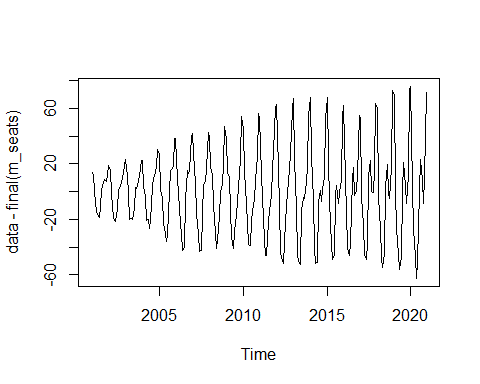
summary(m\_x11)

##   
## Call:  
## seas(x = data, regression.aictest = NULL, x11 = "")  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## MA-Nonseasonal-01 -0.47429 0.05798 -8.180 2.85e-16 \*\*\*  
## MA-Seasonal-12 -0.53983 0.05679 -9.506 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## X11 adj. ARIMA: (0 1 1)(0 1 1) Obs.: 240 Transform: none  
## AICc: 1279, BIC: 1289 QS (no seasonality in final):92.12 \*\*\*  
## Box-Ljung (no autocorr.): 21.33 Shapiro (normality): 0.9955   
## Messages generated by X-13:  
## Warnings:  
## - Automatic transformation selection cannot be done on a  
## series with zero or negative values.

plot(data,col="green",ylim=c(-10,2000),ylab="")  
par(new=T)  
plot(final(m\_x11),col="red",ylim=c(-10,2000),ylab="")  
par(new=T)  
plot(final(m\_seats),col="blue",ylim=c(-10,2000),ylab="")  
legend("topleft",c("Data","X11","SEATS"),col=c("green","red","blue"),lty=c(1,1,1))



# Messing things up after here.  
plot(data-final(m\_seats))



emmm, I am confused that why the result from the *SEATS* is different from that out of *X-11*. The model from X-11 is but SEATS gives me . The former one is right and the corresponding estimation of is also close.

# SEATS

See above.

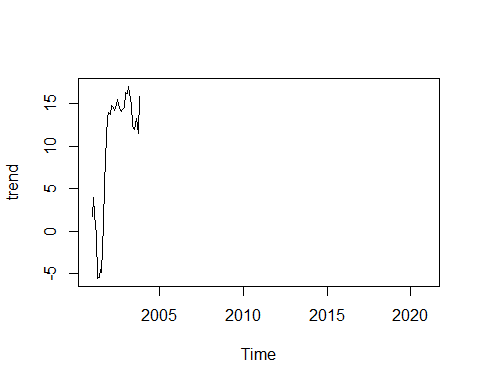
# State Space Model

I just spent a whole day working on the state space model, trying to figure out the steps in this process. To be honest, I was not very, like 80%, clear about bootstrap particle filter, although I spent much time on it during last term. But I have to admit the effort I paid before is worthy, otherwise I can’t come up with the following code within one day. The resource is mainly from **BOOK Monte Carlo Statistical Methods** and the [lecture](https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-842X.00104).  
*题外话：刚刚基本一个半小时的时间内我都在处理转pdf的bug，但是目前这个对我来说并不是很重要。只希望自己以后做事情，要知道自己的主要目的是什么，不要因为一些外在的小事情而耽误太长时间，最终得不偿失。*

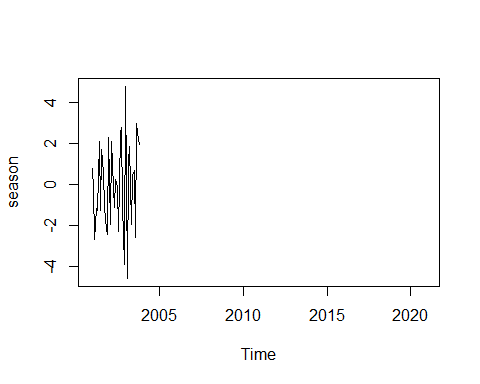
library(forecast)  
set.seed(1)  
model <- Arima(ts(rnorm(24000),freq=12), order=c(0,1,1), seasonal=c(0,1,1),fixed=c(theta=0.5, Theta=0.5))  
data <- simulate(model,nsim=240)  
# let's try SIS(Sequential Importance Sampling) at first  
# Initialization  
Tr\_0 <- rep(rnorm(1),100)  
Se\_0 <- rep(rnorm(1),100)  
w\_0 <- rep(1/100,100)  
  
# Generate particles  
Tr\_1 <- Tr\_0 + rnorm(100)  
Se\_1 <- -Se\_0 + rnorm(100)  
  
# Compute weights  
w\_1 <- w\_0 \* dnorm(rep(data[1],100) - Tr\_1 - Se\_1)  
w\_1 <- w\_1 / sum(w\_1) # normalize  
  
# compute state value  
T\_1 <- sum(w\_1 \* Tr\_1)  
S\_1 <- sum(w\_1 \* Se\_1)

这里，我们只把t\_1时刻的trend和seasonal摘了出来。我们最终的目的是把整个过程（240个时刻）的trend和seasonal求出来。

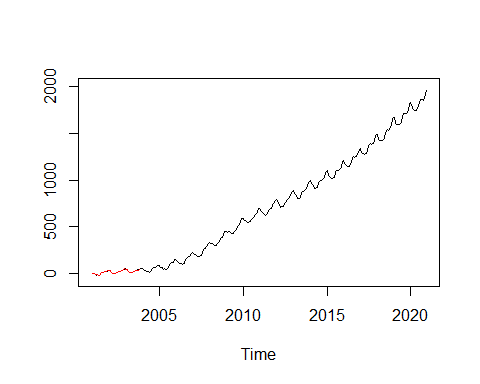
set.seed(1)  
S <- matrix(0,100,251) # Cause Seasonal component's state space model, we have additional 11 zero-values.  
Tr <- matrix(0,100,251)  
S[,12] <- S\_1  
Tr[,12] <- T\_1  
w <- w\_1  
component <- c(T\_1,S\_1)  
for (i in 13:251) {  
 # update particles  
 Tr[,i] <- Tr[,i-1] + rnorm(100)  
 for (j in 1:11) S[,i] <- S[,i]-S[,i-j]  
 S[,i] <- S[,i] + rnorm(100)  
   
 # update weights  
 w <- w \* dnorm(rep(data[i-11],100)-Tr[,i]-S[,i])  
 w <- w / sum(w)  
   
 # evaluate state value  
 t <- sum(w \* Tr[,i])  
 s <- sum(w \* S[,i])  
   
 # add to our component path  
 component <- rbind(component, c(t,s))  
}  
  
trend <- ts(component[,1],frequency = 12, start = c(2001,1),end = c(2020,12))  
season <- ts(component[,2],frequency = 12, start = c(2001,1),end = c(2020,12))  
plot(trend)



plot(season)



plot(data,type="l",ylab="", ylim=c(-50,2000))  
lines(data-season, ylab="",ylim=c(-50,2000),col="red")

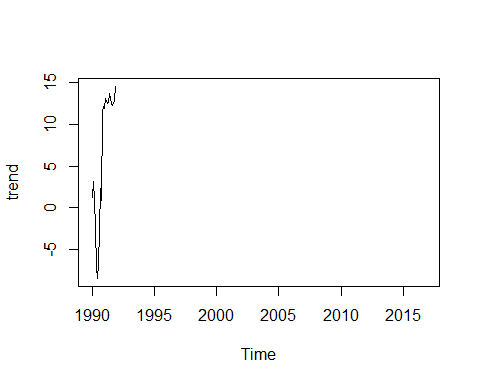


Well, I think something is wrong here again! it’s normal to show some degeneracy here, but the curve after adjusting does not look like that.  
I am going to try it on the data set **unemp**:

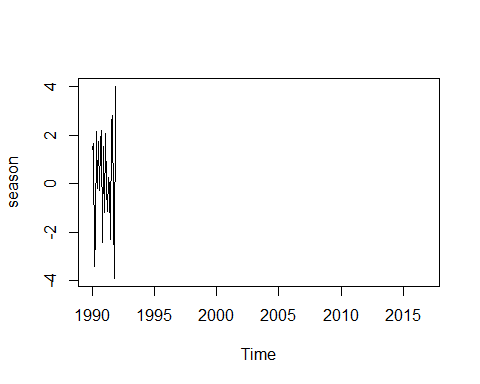
library("seasonal")  
Data <- unemp # 1990.1-2016.11 # 323 records  
summary(Data)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 5153 7012 8078 8766 9476 16147

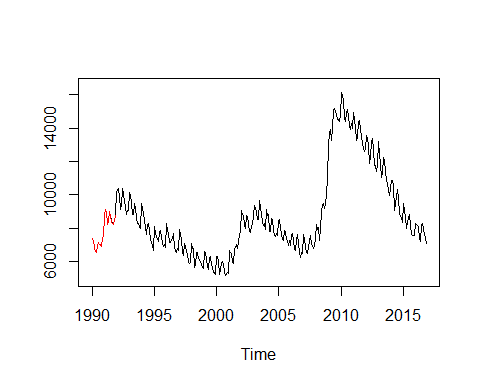
set.seed(1)  
S <- matrix(0,100,334) # Cause Seasonal component's state space model, we have additional 11 zero-values.  
Tr <- matrix(0,100,334) # Hence 323+11 columns  
S[,11] <- rep(0,100)  
Tr[,11] <- rep(0,100)  
w <- rep(1/100,100)  
component <- c()  
for (i in 12:334) {  
 # update particles  
 Tr[,i] <- Tr[,i-1] + rnorm(100)  
 for (j in 1:11) S[,i] <- S[,i]-S[,i-j]  
 S[,i] <- S[,i] + rnorm(100)  
   
 # update weights  
 w <- w \* dnorm(rep(data[i-11],100)-Tr[,i]-S[,i])  
 w <- w / sum(w)  
   
 # evaluate state value  
 t <- sum(w \* Tr[,i])  
 s <- sum(w \* S[,i])  
   
 # add to our component path  
 component <- rbind(component, c(t,s))  
}  
trend <- ts(component[,1],frequency = 12, start = c(1990,1),end = c(2016,11))  
season <- ts(component[,2],frequency = 12, start = c(1990,1),end = c(2016,11))  
plot(trend)



plot(season)



plot(Data,type="l",ylab="", ylim=c(5000,16500))  
lines(Data-season, ylab="",ylim=c(5000,16500),col="red")



The result is still not ideal. I think the initialization is not appropriate, since the trend and seasonal values are too small. I am going to test my guess by using **SIR(Sequential Importance Resampling)**. Since in SIR, there should not have ‘degeneracy’ phenomenon, so if the result is still like the pics above, there must be something wrong.(Use **unemp** data)

library("seasonal")  
Data <- unemp # 1990.1-2016.11 # 323 records  
summary(Data)  
set.seed(1)  
S <- matrix(0,100,334) # Cause Seasonal component's state space model, we have additional 11 zero-values.  
Tr <- matrix(0,100,334) # Hence 323+11 columns  
component <- c()  
for (i in 12:334) {  
 # update particles  
 Tr[,i] <- Tr[,i-1] + rnorm(100)  
 for (j in 1:11) S[,i] <- S[,i]-S[,i-j]  
 S[,i] <- S[,i] + rnorm(100)  
   
 # update weights  
 w <- dnorm(rep(Data[i-11],100)-Tr[,i]-S[,i])  
 w <- w / sum(w)  
   
 # evaluate state value  
 t <- sum(w \* Tr[,i])  
 s <- sum(w \* S[,i])  
   
 # add to our component path  
 component <- rbind(component, c(t,s))  
   
 # resample  
 Tr[,i] <- sample(Tr[,i], size=100, replace=TRUE, prob=w)  
 S[,i] <- sample(S[,i], size=100, replace=TRUE, prob=w)  
}  
trend <- ts(component[,1],frequency = 12, start = c(1990,1),end = c(2016,11))  
season <- ts(component[,2],frequency = 12, start = c(1990,1),end = c(2016,11))  
plot(trend)  
plot(season)  
plot(Data,type="l",ylab="", ylim=c(5000,16500))  
lines(Data-season, ylab="",ylim=c(5000,16500),col="red")

library("seasonal")  
Data <- unemp # 1990.1-2016.11 # 323 records  
summary(Data)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 5153 7012 8078 8766 9476 16147

set.seed(1)  
S <- matrix(0,100,334) # Cause Seasonal component's state space model, we have additional 11 zero-values.  
Tr <- matrix(0,100,334) # Hence 323+11 columns  
component <- c()  
for (i in 12:334) {  
 # update particles  
 Tr[,i] <- Tr[,i-1] + rnorm(100)  
 for (j in 1:11) S[,i] <- S[,i]-S[,i-j]  
 S[,i] <- S[,i] + rnorm(100)  
 # update weights  
 w <- dnorm(rep(Data[i-11],100)-Tr[,i]-S[,i])  
 w <- w / sum(w)  
}  
dnorm(rep(Data[323],100)-Tr[,334]-S[,334])

## [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

# update weights  
 w <- dnorm(rep(Data[1],100)-Tr[,12]-S[,12])  
 w <- w / sum(w)

**- Update 6.2 -** 从上次和Aaron见完之后，就一直在划水，心里也挺不过意不去的。因为种种事情都没有静下心来学习，感觉今天积攒了很久的洪荒之力可以稍微发泄一下？另外就是手腕这两天也一直没有休息过来，还是要多加注意呀。接下来说正事儿吧。

If we analyse the case above why the weights are zeros, it is not difficult to find our initializations are around 0 and , our data is quite large. If we use to generate our weights, it is obvious that weights are gonna very small, almost 0.( is the density of standard normal distribution.)  
So why don’t I let the first trend component (i.e. ) to be equal to ? Cause we suppose

well,no more talking, let’s try. (Hope it works well)

library("seasonal")  
Data <- unemp # 1990.1-2016.11 # 323 records  
summary(Data)  
set.seed(1)  
S <- matrix(0,100,334) # Cause Seasonal component's state space model, we have additional 11 zero-values.  
Tr <- matrix(0,100,334) # Hence 323+11 columns  
Tr[,11] <- Data[1]  
component <- c()  
for (i in 12:334) {  
 # update particles  
 Tr[,i] <- Tr[,i-1] + rnorm(100)  
 for (j in 1:11) S[,i] <- S[,i]-S[,i-j]  
 S[,i] <- S[,i] + rnorm(100)  
   
 # update weights  
 w <- dnorm(rep(Data[i-11],100)-Tr[,i]-S[,i])  
 w <- w / sum(w)  
   
 # evaluate state value  
 t <- sum(w \* Tr[,i])  
 s <- sum(w \* S[,i])  
   
 # add to our component path  
 component <- rbind(component, c(t,s))  
   
 # resample  
 Tr[,i] <- sample(Tr[,i], size=100, replace=TRUE, prob=w)  
 S[,i] <- sample(S[,i], size=100, replace=TRUE, prob=w)  
}  
trend <- ts(component[,1],frequency = 12, start = c(1990,1),end = c(2016,11))  
season <- ts(component[,2],frequency = 12, start = c(1990,1),end = c(2016,11))  
plot(trend)  
plot(season)  
plot(Data,type="l",ylab="", ylim=c(5000,16500),lwd=1.5)  
lines(Data-season, ylab="",ylim=c(5000,16500),col="red")

This chunk is wrong since the weights’ problem. I am going to check the weights below:

set.seed(1)  
S <- matrix(0,100,334) # Cause Seasonal component's state space model, we have additional 11 zero-values.  
Tr <- matrix(0,100,334) # Hence 323+11 columns  
Tr[,11] <- Data[1]  
  
for (i in 12) {  
   
 # update particles  
 Tr[,i] <- Tr[,i-1] + rnorm(100)  
 for (j in 1:11) {  
 S[,i] <- S[,i]-S[,i-j]   
 }  
 S[,i] <- S[,i] + rnorm(100)  
   
 # update weights  
 w <- dnorm(Data[i-11]-Tr[,i]-S[,i])  
 w <- w / sum(w)  
}  
  
Tr[,13] <- Tr[,12] +rnorm(100)  
S[,13] = -S[,12] + rnorm(100)  
w<-dnorm(Data[2]-Tr[,13]-S[,13])  
w

## [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Data[2]-Tr[,13]-S[,13]

## [1] -118.2970 -117.7831 -120.6332 -117.7227 -117.3530 -118.4221 -117.5208  
## [8] -117.9367 -116.1536 -115.8456 -120.0268 -118.3713 -114.0921 -113.4100  
## [15] -119.2842 -119.9368 -116.5111 -116.8486 -116.6751 -118.2208 -117.9996  
## [22] -117.9383 -117.4735 -112.9794 -117.0455 -118.4905 -116.2332 -114.1319  
## [29] -116.5367 -118.6846 -116.9856 -113.0281 -115.8482 -120.3995 -114.9224  
## [36] -118.7560 -118.4141 -116.2869 -119.4966 -117.2850 -120.1967 -115.5407  
## [43] -119.8316 -116.8520 -116.3252 -113.6343 -115.6486 -117.3042 -118.8111  
## [50] -122.4297 -118.5580 -117.4910 -117.9695 -116.3595 -122.1939 -121.7981  
## [57] -112.0561 -116.9942 -117.4100 -114.3751 -117.3352 -118.1242 -115.7114  
## [64] -116.5690 -118.4263 -115.8650 -115.7062 -119.2689 -116.3941 -116.6620  
## [71] -114.5199 -116.8565 -115.4788 -118.7234 -115.0687 -118.4481 -113.6088  
## [78] -115.5210 -114.5668 -117.8859 -118.8234 -116.5848 -118.3592 -116.4068  
## [85] -117.6471 -116.4341 -113.8010 -118.7519 -116.3290 -118.3872 -117.5685  
## [92] -118.7979 -120.2280 -115.7502 -121.4632 -117.8166 -116.6408 -118.4427  
## [99] -114.5977 -117.4671

…stupid… The result is still not right. The assumption just worked at the first moment then failed.

Maybe I can use the first twelve values to generate an initial set of seasonal components? In other words, i) compute the average ; ii) use the differences as seasonal components of one loop. But this will be only correct when there is no trend influence in this ts!

**WHAT CAN I DO ?** 自己再多想一想 总会有办法的。网上查查资料吧。

Wait… 我为什么不试一试我自己的数据呢？如果trend是缓慢变化的话(*-Update 6.4- actually I am not sure whether we can use the state space model above in* ***unemp*** *data set, since the sarima model of it is different from that of Durbin’s report*)，某种程度上SIR应该是可以用的吧？ try it:

set.seed(1)  
S <- matrix(0,100,251) # Cause Seasonal component's state space model, we have additional 11 zero-values.  
Tr <- matrix(0,100,251)  
Tr[,11] <- data[1]  
component <- c()  
for (i in 12:251) {  
   
 # update particles  
 Tr[,i] <- Tr[,i-1] + rnorm(100)  
 for (j in 1:11) S[,i] <- S[,i]-S[,i-j]  
 S[,i] <- S[,i] + rnorm(100)  
   
 # update weights  
 w <- dnorm(data[i-11]-Tr[,i]-S[,i])  
 w <- w / sum(w)  
   
 # evaluate state value  
 t <- sum(w \* Tr[,i])  
 s <- sum(w \* S[,i])  
   
 # add to our component path  
 component <- rbind(component, c(t,s))  
   
 # resample  
 Tr[,i] <- sample(Tr[,i], size =100, replace = TRUE, prob = w)  
 S[,i] <- sample(S[,i], size = 100, replace = TRUE, prob = w)  
}  
  
trend <- ts(component[,1],frequency = 12, start = c(2001,1),end = c(2020,12))  
season <- ts(component[,2],frequency = 12, start = c(2001,1),end = c(2020,12))  
plot(trend)  
plot(season)  
plot(data,type="l",ylab="", ylim=c(-50,2000))  
lines(data-season, ylab="",ylim=c(-50,2000),col="red")

The chunk above can not run. The code below is to check where the degeneracy happens:

set.seed(1)  
S <- matrix(0,100,251) # Cause Seasonal component's state space model, we have additional 11 zero-values.  
Tr <- matrix(0,100,251)  
Tr[,11] <- data[1]  
component <- c()  
for (i in 12:14) {  
   
 # update particles  
 Tr[,i] <- Tr[,i-1] + rnorm(100)  
 for (j in 1:11) S[,i] <- S[,i]-S[,i-j]  
 S[,i] <- S[,i] + rnorm(100)  
   
 # update weights  
 w <- dnorm(data[i-11]-Tr[,i]-S[,i])  
 w <- w/sum(w)  
   
 # evaluate state value  
 t <- sum(w \* Tr[,i])  
 s <- sum(w \* S[,i])  
   
 # add to our component path  
 component <- rbind(component, c(t,s))  
   
 # resample  
 Tr[,i] <- sample(Tr[,i], size =100, replace = TRUE, prob = w)  
 S[,i] <- sample(S[,i], size = 100, replace = TRUE, prob = w)  
}  
w

## [1] 2.189091e-24 2.326805e-34 1.453344e-22 2.441187e-09 2.014216e-08  
## [6] 1.762221e-03 4.396355e-05 7.945319e-11 3.757556e-18 4.418994e-36  
## [11] 1.073132e-21 1.629745e-30 1.429414e-05 3.265350e-10 2.720454e-20  
## [16] 5.763297e-06 9.120677e-18 9.231106e-12 7.140953e-20 5.607044e-11  
## [21] 4.899712e-06 5.591104e-07 7.726781e-28 4.144347e-25 2.530796e-18  
## [26] 3.589514e-15 4.510176e-10 5.292965e-09 1.169976e-09 2.031101e-21  
## [31] 2.325723e-08 1.059966e-09 8.206373e-03 1.149813e-12 1.629605e-16  
## [36] 1.070495e-24 7.651830e-18 2.495841e-25 1.838943e-09 1.159049e-20  
## [41] 4.345031e-03 4.117133e-11 2.436136e-30 2.867601e-18 4.302048e-15  
## [46] 7.983313e-13 4.773519e-14 3.259247e-09 9.021973e-22 3.200885e-21  
## [51] 1.028877e-10 6.213282e-09 3.301319e-12 1.386625e-18 1.950140e-13  
## [56] 1.764682e-02 1.479019e-35 7.262959e-20 5.795653e-12 3.308094e-12  
## [61] 4.730831e-16 1.791086e-14 1.514309e-07 5.856578e-12 2.038496e-13  
## [66] 2.313757e-24 6.958226e-11 2.208192e-27 3.243754e-29 2.250399e-19  
## [71] 1.440509e-01 3.880587e-10 2.857292e-15 5.844375e-10 1.335457e-28  
## [76] 1.773912e-11 1.298437e-14 2.161121e-26 3.018623e-13 5.813938e-12  
## [81] 3.499301e-12 7.486624e-09 1.720131e-18 5.330352e-29 2.705016e-29  
## [86] 9.156298e-06 1.859245e-17 1.903414e-19 6.750105e-11 3.220691e-22  
## [91] 1.191600e-10 3.542694e-14 2.149015e-09 9.207351e-28 2.169345e-16  
## [96] 1.917820e-17 8.239033e-01 6.491148e-06 1.056944e-15 3.806593e-14

# seperate   
w <- dnorm(data[3]-Tr[,14]-S[,14])  
w <- w/sum(w)  
w

## [1] 1.101409e-02 1.101409e-02 1.101409e-02 5.144544e-06 5.144544e-06  
## [6] 1.101409e-02 1.101409e-02 1.101409e-02 1.101409e-02 1.101409e-02  
## [11] 1.101409e-02 1.101409e-02 1.101409e-02 1.101409e-02 5.144544e-06  
## [16] 1.925698e-03 1.101409e-02 1.101409e-02 1.101409e-02 5.144544e-06  
## [21] 1.101409e-02 1.101409e-02 3.477036e-02 1.167785e-02 1.101409e-02  
## [26] 5.144544e-06 1.101409e-02 1.925698e-03 1.101409e-02 1.101409e-02  
## [31] 6.619043e-03 1.101409e-02 1.101409e-02 1.101409e-02 1.101409e-02  
## [36] 1.101409e-02 1.101409e-02 1.101409e-02 1.101409e-02 1.101409e-02  
## [41] 3.477036e-02 1.841524e-02 5.144544e-06 1.101409e-02 1.101409e-02  
## [46] 1.101409e-02 1.101409e-02 3.477036e-02 1.101409e-02 5.144544e-06  
## [51] 1.101409e-02 1.101409e-02 1.101409e-02 1.101409e-02 2.781106e-05  
## [56] 1.101409e-02 1.101409e-02 1.101409e-02 1.101409e-02 1.101409e-02  
## [61] 1.101409e-02 1.101409e-02 5.144544e-06 1.101409e-02 5.144544e-06  
## [66] 5.144544e-06 1.101409e-02 1.101409e-02 3.477036e-02 1.101409e-02  
## [71] 1.925698e-03 5.144544e-06 2.781106e-05 1.101409e-02 3.477036e-02  
## [76] 1.101409e-02 1.101409e-02 5.144544e-06 1.101409e-02 1.101409e-02  
## [81] 4.989486e-05 1.101409e-02 1.101409e-02 5.144544e-06 4.137657e-03  
## [86] 5.144544e-06 1.101409e-02 5.144544e-06 1.101409e-02 6.619043e-03  
## [91] 1.101409e-02 1.101409e-02 5.144544e-06 1.101409e-02 3.477036e-02  
## [96] 1.101409e-02 1.101409e-02 1.101409e-02 1.101409e-02 1.101409e-02

It is very weird. The result from the for loop is different from that of separate code.  
**-Update 6.4-** after talking with Aaron, I realized where I made a mistake. The Tr and S I used in the seperate case is after resampling…(stupid -\_-|)

——————————–**UPDATE 6.4**————————————-