---

title: "BuildThePrior"

author: "Linyi Guo"

date: "2020/1/8"

output:

html\_document:

toc: true

---

```{r include=FALSE}

set.seed(9483)

rm(list=ls())

```

```{r}

# install these packages if needed

#install.packages("seasonal")

#install.packages("forecast")

#install.packages("KFAS")

# import package

library(seasonal)

library(forecast)

library(KFAS)

```

# SIMULATION

To avoid potential mistakes, I only simulate the datasets from ARIMA(1,1,1)(0,1,1)

## Define a function to simulate data that we want

```{r}

simulation1 <- function(length){

model <- Arima(ts(rnorm(120),start=c(1980,01),frequency =12), order=c(1,1,1),

seasonal=c(0,1,1), fixed=c(phi=runif(1), theta=runif(1),

Theta=runif(1))

)

data <- simulate(model, nsim=length)

# because if we need to take log later, data must be positive

if(min(data) <= 0) data <- data - min(data) + runif(1)

else data <- data

return(data)

}

```

## function to simulate a datalist with a lot of datasets

```{r}

simlist1 <- function(n,length) {

Datalist <- list()

for (i in 1:n) Datalist[[i]] <- simulation1(length)

return(Datalist)

}

```

```{r}

set.seed(9483)

datalist2 <- simlist1(100, 180)

# If time permits, you can try this one

# datalist2 <- simlist1(200, 180)

```

Until this step, datalist is basically what I want to use in the following analysis, but we still have one more step: that is to preprocessing.

# Preprocessing

```{r}

# define a function for outliers and log-transformation

preprocess <- function(x11) {

if(transformfunction(x11) == 'log')

data <- log(series(x11, 'b1'))

else

data <- series(x11, 'b1')

return(data)

}

```

```{r}

# build model list

x11list2 <- lapply(datalist2, function(x) seas(x, x11=''))

```

```{r}

# obtain the preprocessed datalist denoted as Datalist

Datalist2 <- lapply(x11list2, preprocess)

```

Now we finished this step, then we need to find the 'ideal' values of parameters in these datasets.

# Building Our Prior

## Searching for the 'best' value

\*\*Loss function\*\*

```{r}

Dif1 <- function(x11, ssm, data, sigma){

x11\_trend <- series(x11, 'd12')

x11\_seasonal <- series(x11, 'd10')

x11\_irregular <- series(x11, 'd13')

ssm\_trend <- coef(ssm, states = 'trend')

ssm\_seasonal <- -rowSums(coef(ssm, states='seasonal'))

ssm\_irregular <- data[-1] - ssm\_trend[-1] - ssm\_seasonal[-length(data)]

D <- sum((x11\_irregular[-1]-ssm\_irregular)^2)/sigma[1] +

sum((x11\_trend-ssm\_trend)^2)/sigma[2] +

sum((x11\_seasonal[-1]-ssm\_seasonal[-length(data)])^2)/sigma[3]

return(D)

}

```

\*\*Exhaustion function\*\*

```{r}

exhaustion1 <- function(data){

Difference <- c()

index <- c()

x11 <- seas(data, x11='')

for (i in 1:100) {

for (j in 1:100) {

ssmm <- SSModel(data ~ SSMtrend(1, Q=list(j\*0.2)) +

SSMseasonal(12, sea.type = 'dummy', Q = 1),

H = i\*0.2)

ssm <- KFS(ssmm)

sigma <- c(i\*0.2, j\*0.2, 1)

dif <- Dif1(x11, ssm, data, sigma)

Difference <- c(Difference, dif)

index <- rbind(index, sigma)

}

}

df <- data.frame(variance=index, difference = Difference)

return(df)

}

```

```{r}

# system.time(exhaustion1(Datalist1[[1]]))

# user system elapsed

# 189.80 237.43 427.96

idevallist2 <- lapply(Datalist2, exhaustion1)

```

```{r}

idevalmat2 <- c()

for (i in 1:100){

ideval <- idevallist2[[i]][which.min(idevallist2[[i]]$difference),c(1,2)]

idevalmat2 <- rbind(idevalmat2, ideval)

}

```

And then please output the data frame `idevalmat2`.

```

write.csv(idevalmat2, "The position//idevalmat2.csv")

```