San Francisco precipitation

Julien JACQUES

2/19/2020

We extract training and test set

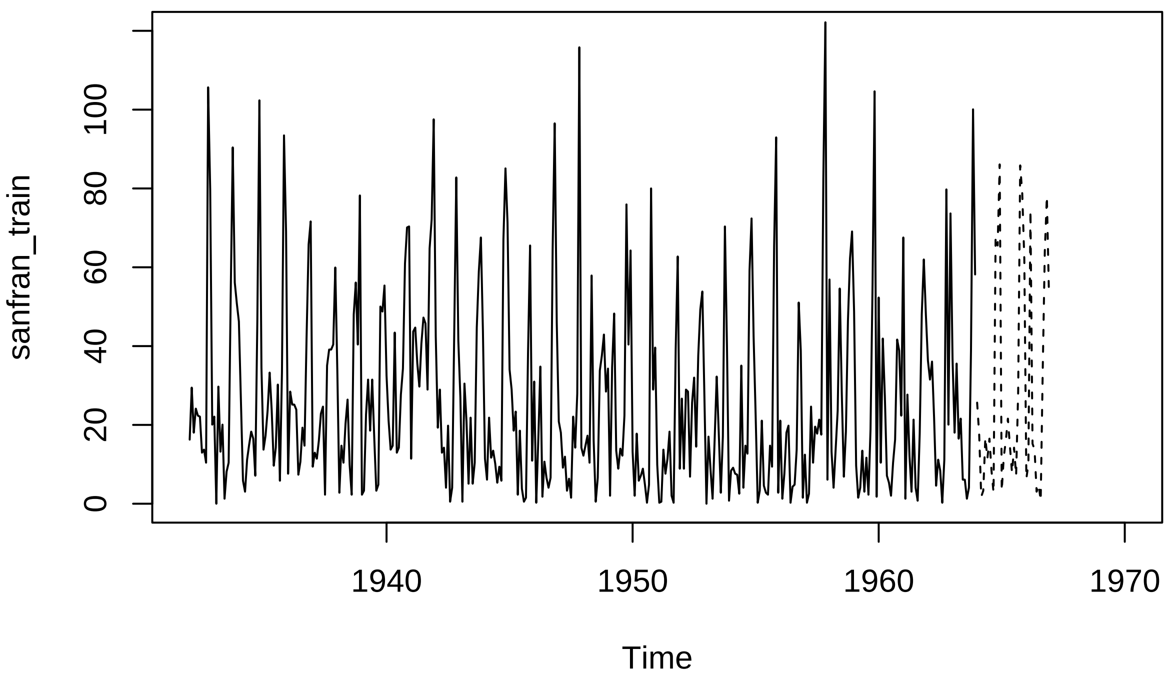
```
data=scan(file="data/sanfran.csv",skip=1)
sanfran<-ts(data,start=c(1932,1),end=c(1966,12),freq=12)
library(forecast)
```

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

```
sanfran_train=window(sanfran,start=c(1932,1),end=c(1963,12))
sanfran_test=window(sanfran,start=c(1964,1),end=c(1966,12))
```

We can plot both

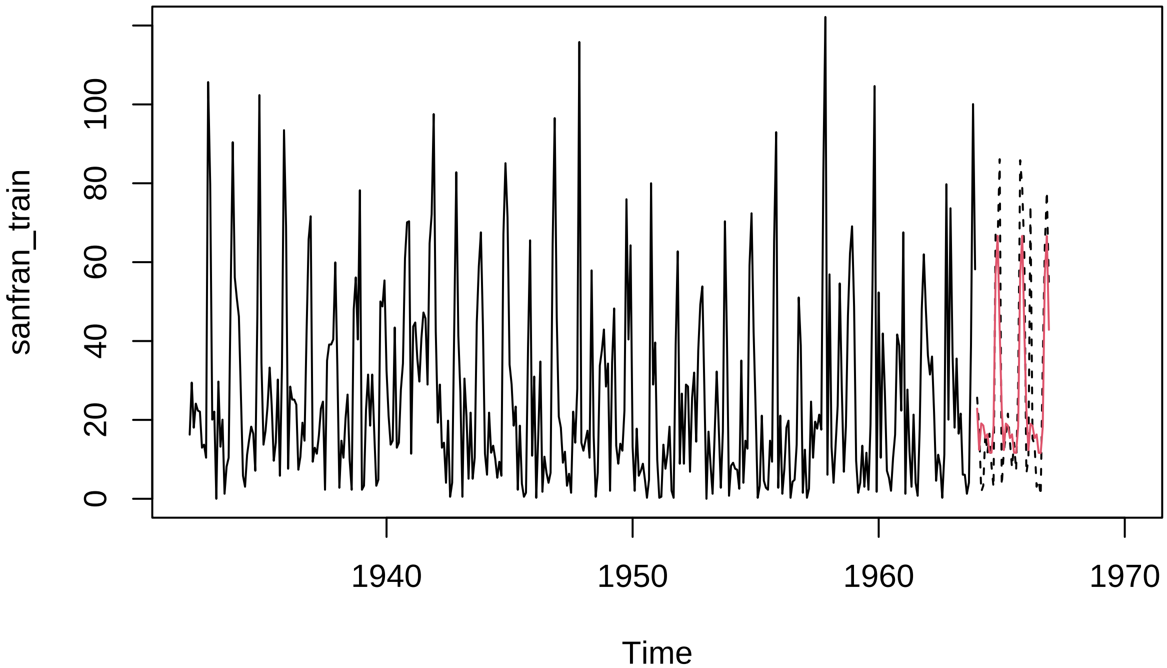
```
plot(sanfran_train,xlim=c(1932,1970),ylim=c(0,120))
lines(sanfran_test,lty=2)
```



Forecasting with exponential smoothing

We see a seasonal pattern, probably additive.

```r
library(forecast)
h=hw(sanfran_train,seasonal='additive',damped=FALSE,h=36)
plot(sanfran_train,xlim=c(1932,1970),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(h$mean,col=2)
```

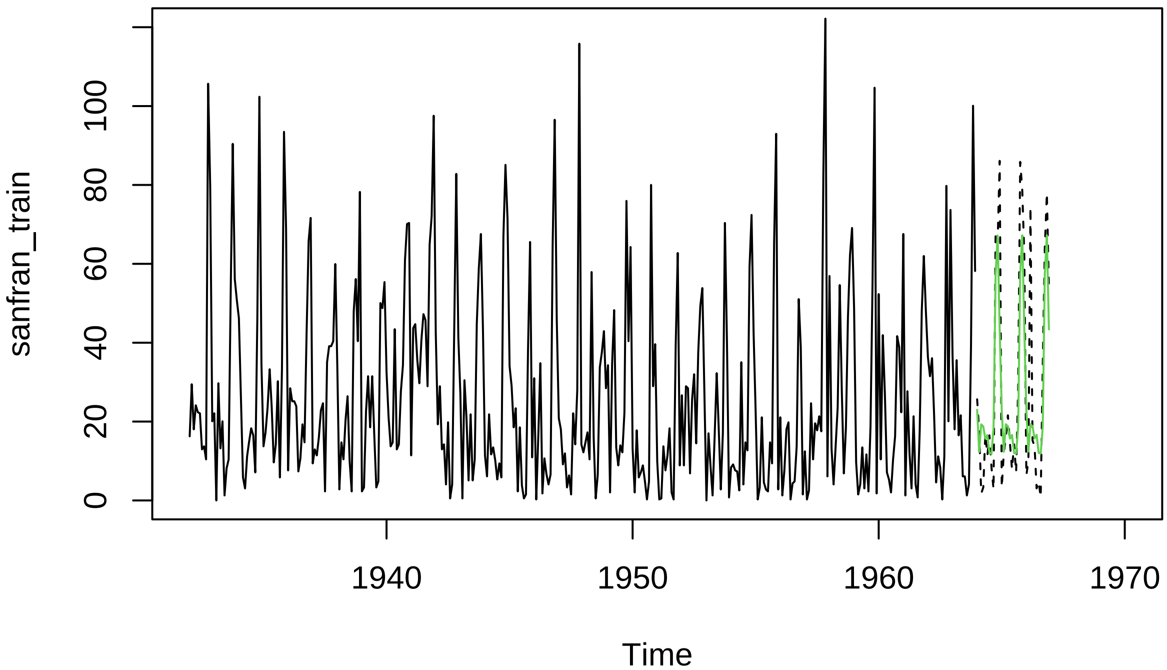



```r
print(sqrt(mean((h$mean-sanfran_test)^2)))
```

```
## [1] 15.86614
```

We can compare with a damped version, the result are slightly better

```r
hd=hw(sanfran_train,seasonal='additive',damped=TRUE,h=36)
plot(sanfran_train,xlim=c(1932,1970),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(hd$mean,col=3)
```

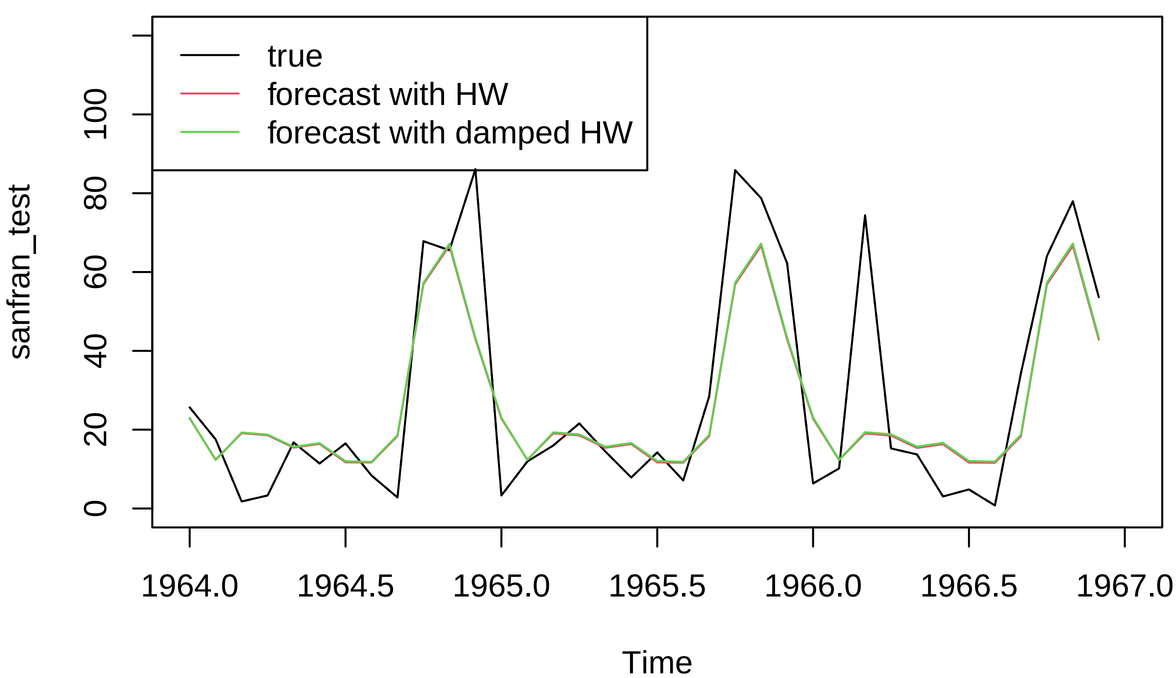



```
print(sqrt(mean((hd$mean-sanfran_test)^2)))
```

```
## [1] 15.77082
```

We can zoom on the prediction

```
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(h$mean,col=2)
lines(hd$mean,col=3)
legend('topleft',col=1:3,lty=1,legend=c('true','forecast with HW','forecast with damped HW'))
```



The difference is almost null between HW and its damped version. Indeed, if we have a look to the ϕ

parameter, it is very close to 1 ($\phi = 0.9725$): the damping effect is almost null.

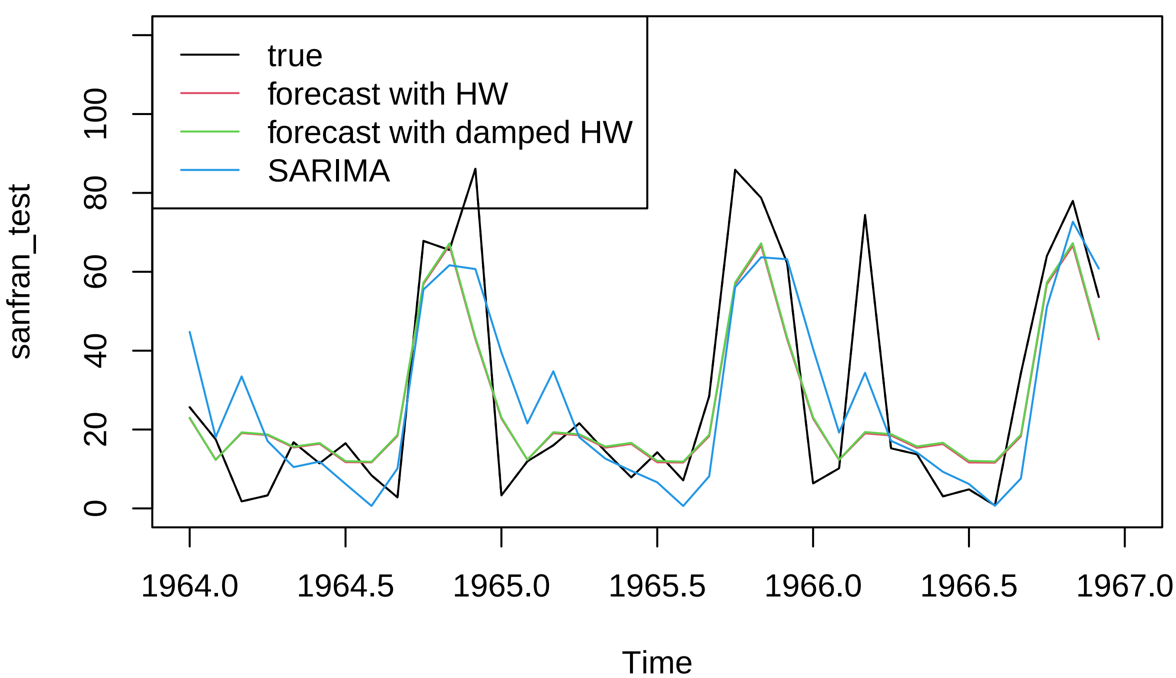
Forecasting with SARIMA

Simple solution: with auto.arima function:

```
fit=auto.arima(sanfran_train)
summary(fit)
```

```
## Series: sanfran_train
## ARIMA(0,0,1)(2,1,0)[12] with drift
##
## Coefficients:
##           ma1     sar1     sar2    drift
##       -0.0108  -0.6204  -0.2710  -0.0061
## s.e.   0.0510   0.0508   0.0521   0.0415
##
## sigma^2 = 327.7:  log likelihood = -1605.73
## AIC=3221.46   AICc=3221.62   BIC=3241.05
##
## Training set error measures:
##                       ME     RMSE      MAE  MPE MAPE      MASE         ACF1
## Training set -0.04091504 17.72204 13.27102 -Inf  Inf 0.8385827 0.0009082269
```

```
prev=forecast(fit,h=36)
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(h$mean,col=2)
lines(hd$mean,col=3)
lines(prev$mean,col=4)
legend('topleft',col=1:4,lty=1,legend=c('true','forecast with HW','forecast with damped HW','SARIMA'))
```



```r
print(sqrt(mean((prev$mean-sanfran_test)^2)))
```

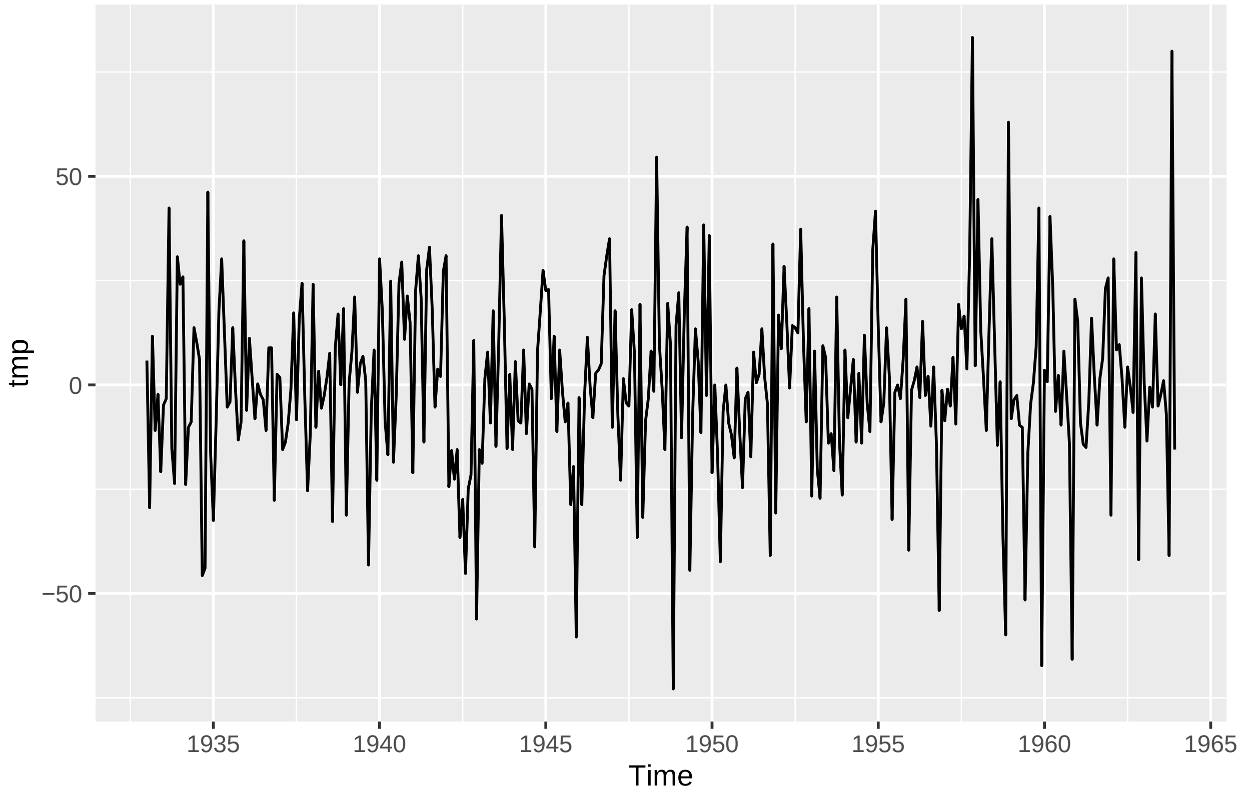
```
## [1] 16.57343
```

The forecast is not better than with HW.

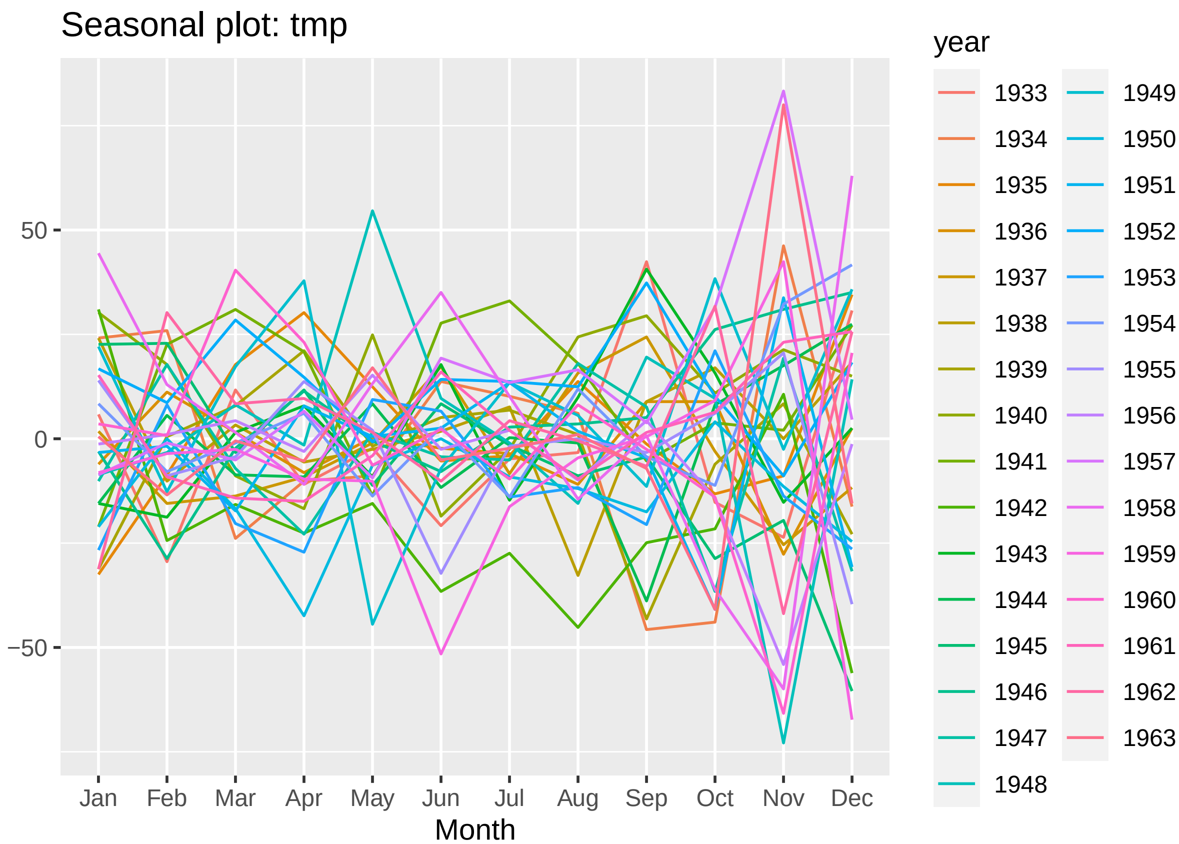
We can try to choose manually the ordre of the SARIMA model.

Let's start by differeciating the serie.

```r
tmp=diff(sanfran_train,lag=12)
autoplot(tmp)
```

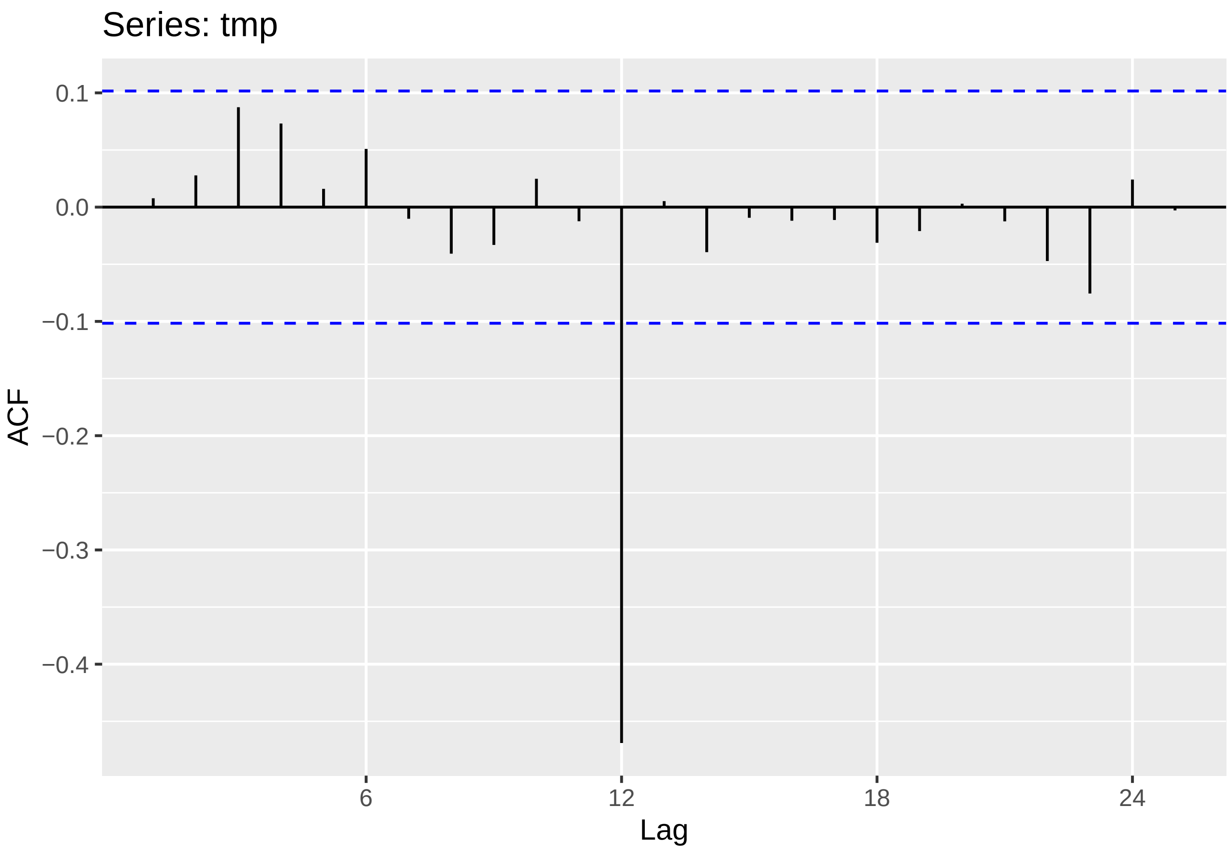



```r
ggseasonplot(tmp)
```

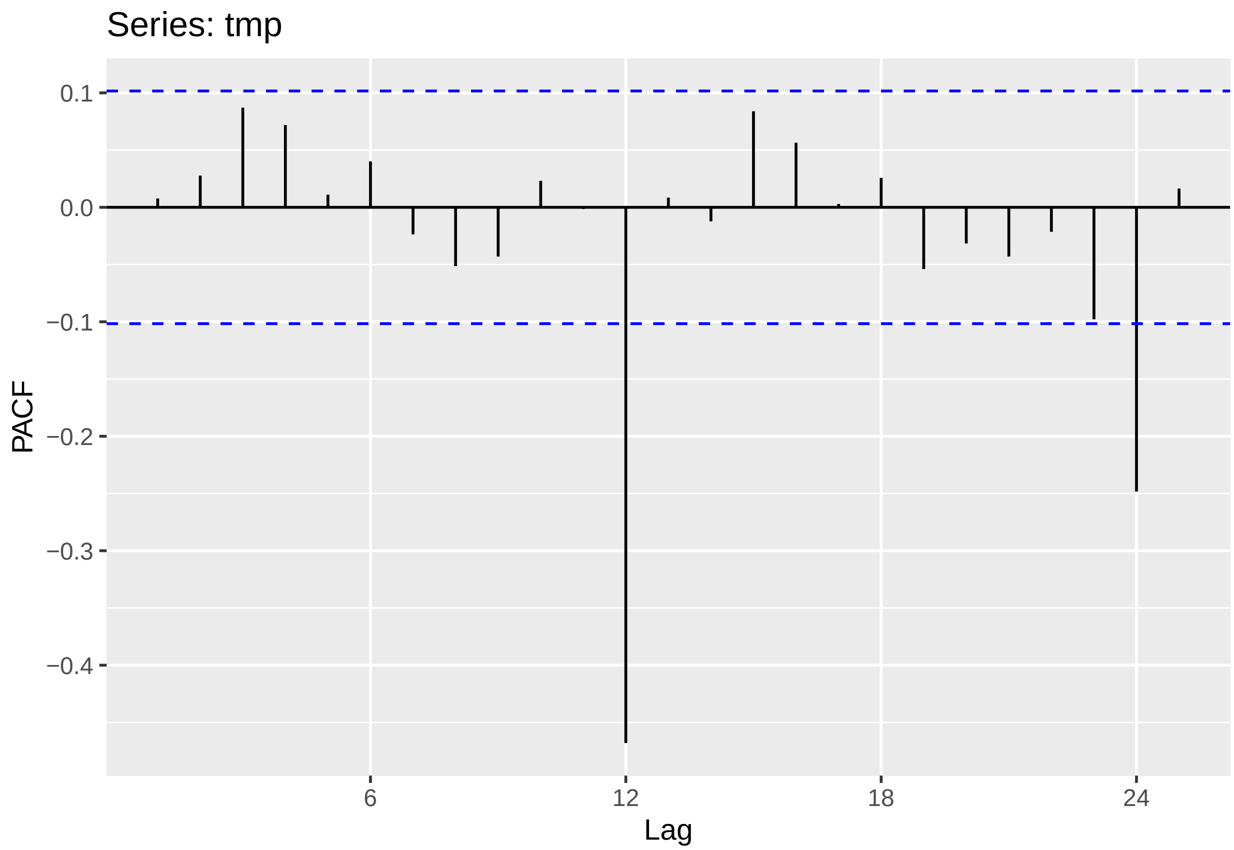


It seems approximatively stationary. Let's look at the ACF and PACF

```
ggAcf(tmp)
```

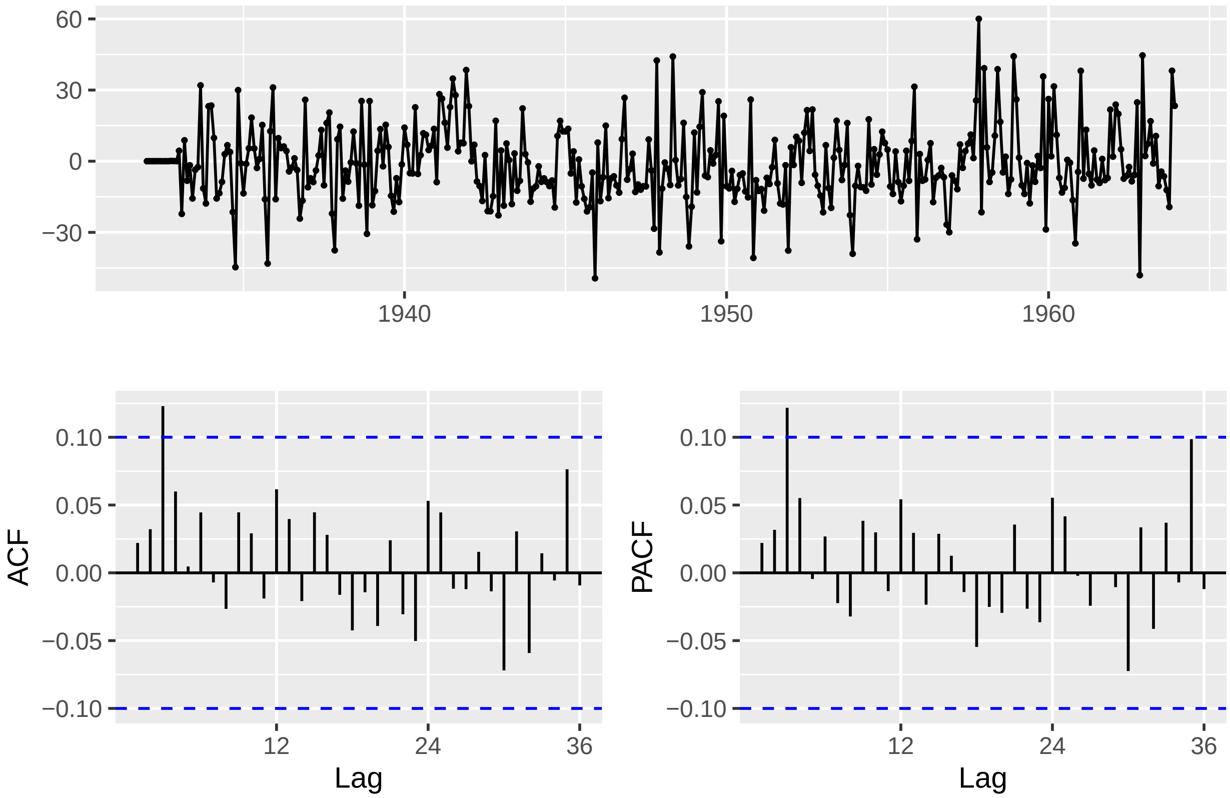



```r
ggPacf(tmp)
```



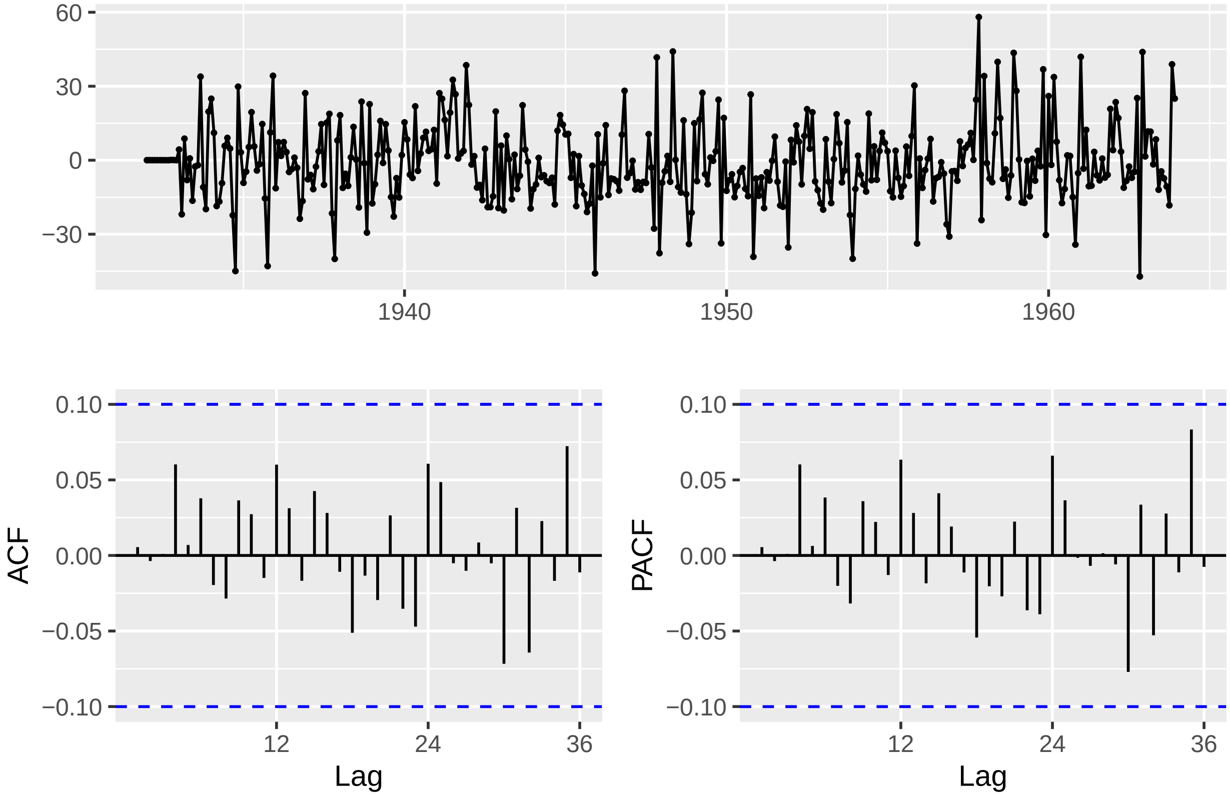
- the significant ACF at lag 12 and the exponential decay of the seasonal lags of the PACF suggest a seasonnal MA_1

```
fit=Arima(sanfran_train, order=c(0,0,0), seasonal=c(0,1,1))
fit %>% residuals() %>% ggtsdisplay()
```



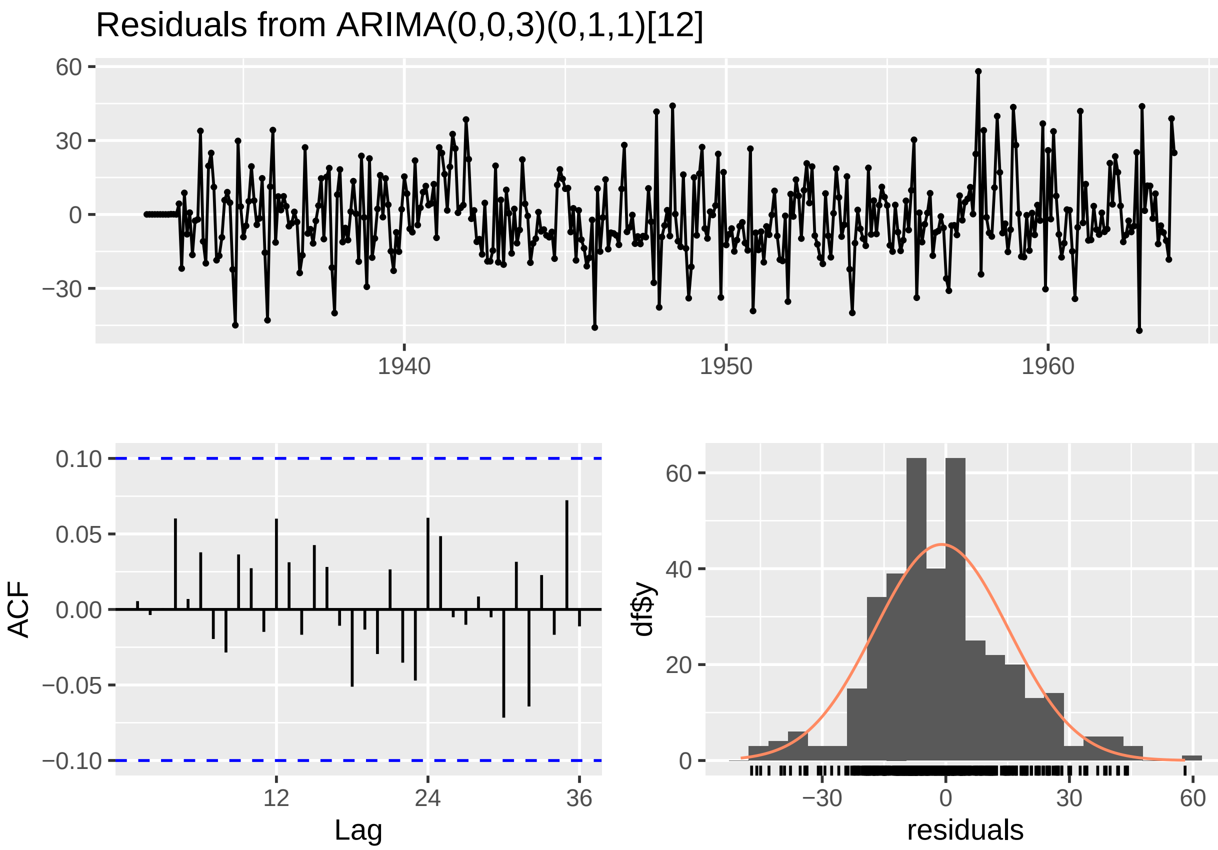
- There is still significant ACF and PACF at lag 3. We can add some additional non-seasonal terms, with an $SARIMA_{(0,0,3)(0,1,1)_{12}}$ (or $SARIMA_{(3,0,0)(0,1,1)_{12}}$)

```
fit1=Arima(sanfran_train, order=c(0,0,3), seasonal=c(0,1,1))
fit1 %>% residuals() %>% ggtsdisplay()
```



It seems that we have captured all auto-correlations

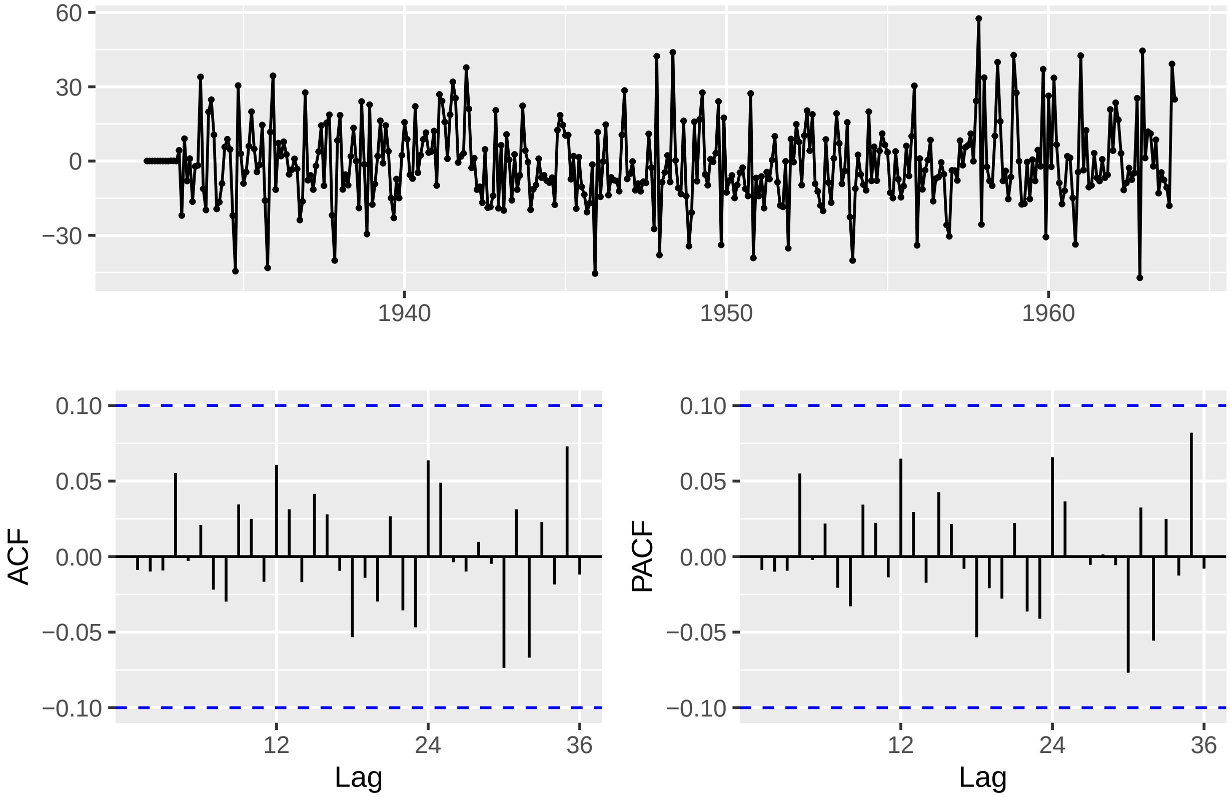
```
checkresiduals(fit1)
```



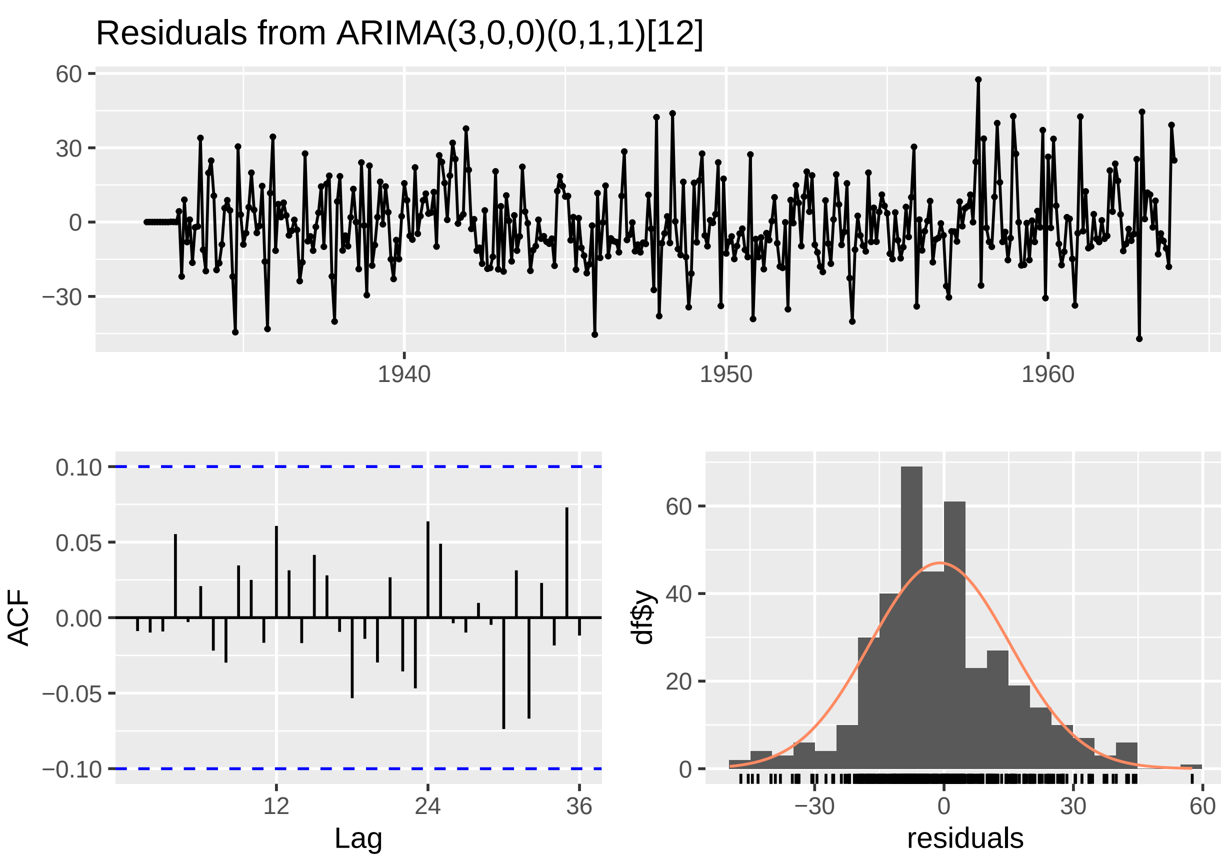

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,3)(0,1,1)[12]
## Q* = 11.139, df = 20, p-value = 0.9425
##
## Model df: 4.   Total lags used: 24
```

We have the same result with an $SARIMA_{(3,0,0)(0,1,1)_{12}}$

```
fit2=Arima(sanfran_train, order=c(3,0,0), seasonal=c(0,1,1))
fit2 %>% residuals() %>% ggtsdisplay()
```




```r
checkresiduals(fit2)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(3,0,0)(0,1,1)[12]
## Q* = 10.834, df = 20, p-value = 0.9504
##
## Model df: 4.   Total lags used: 24
```

Both model are acceptable. We can compare the AICc:

```
cat('AICc for SARIMA_{(0,0,3)(0,1,1)_{12}} : ',fit1$aicc,'\n')
```

```
## AICc for SARIMA_{(0,0,3)(0,1,1)_{12}} :  3171.135
```

```
cat('AICc for SARIMA_{(3,0,0)(0,1,1)_{12}} : ',fit2$aicc,'\n')
```

```
## AICc for SARIMA_{(3,0,0)(0,1,1)_{12}} :  3170.473
```

The second one is better, we select it for forecasting.

We can forecast the next 30 values and compare the results

```
prev=forecast(fit2,h=36)
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(h$mean,col=2)
print(sqrt(mean((h$mean-sanfran_test)^2)))
```

```
## [1] 15.86614
```

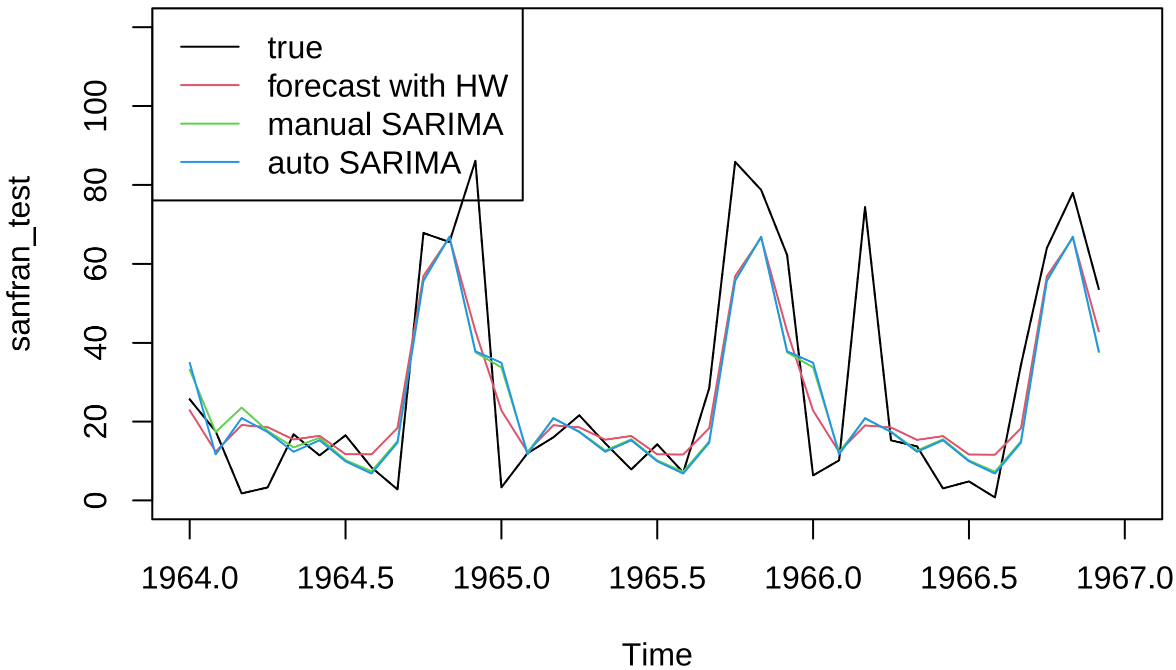
```
lines(prev$mean,col=3)
print(sqrt(mean((prev$mean-sanfran_test)^2)))
```

```
## [1] 17.49685
```

```
prev=forecast(fit,h=36)
lines(prev$mean,col=4)
print(sqrt(mean((prev$mean-sanfran_test)^2)))
```

```
## [1] 17.5374
```

```
legend('topleft',col=1:4,lty=1,legend=c('true','forecast with HW','manual SARIMA', 'auto SARIMA'))
```



The results are not better with this model than with this one selected by auto.arima

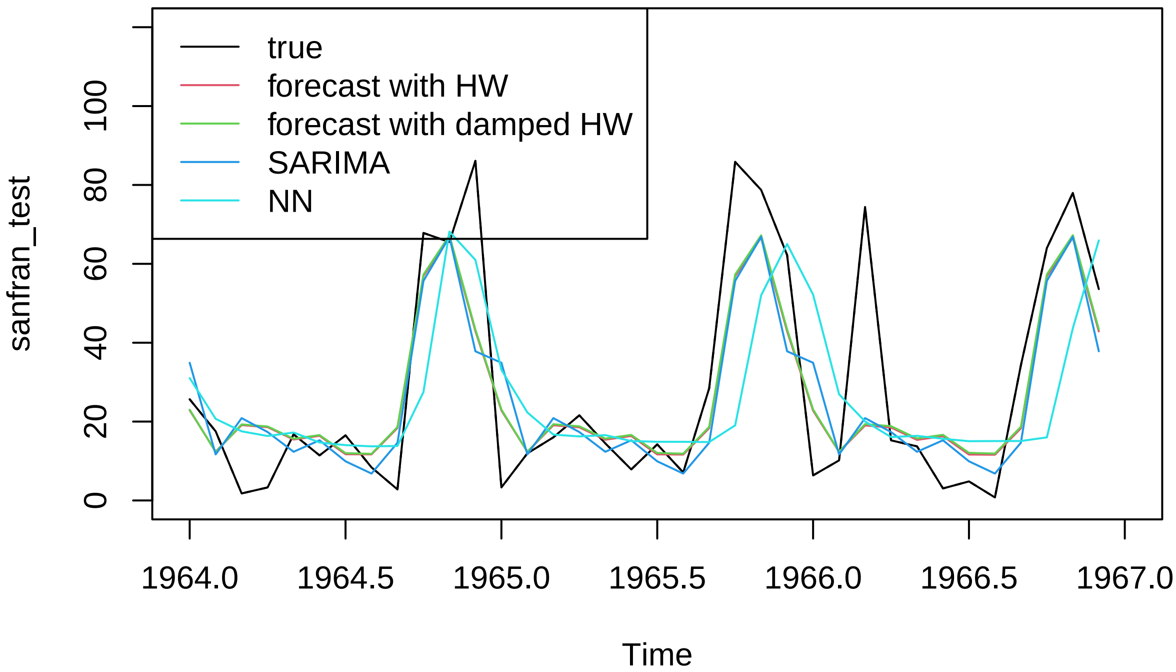
Forecasting with Neural Network

We van automatically select the best $NNAR_{(p,P,k)_T}$:

```
fit=nnetar(sanfran_train)
print(fit)
```

```
## Series: sanfran_train
## Model:  NNAR(4,1,3)[12]
## Call:   nnetar(y = sanfran_train)
##
## Average of 20 networks, each of which is
## a 5-3-1 network with 22 weights
## options were - linear output units
##
## sigma^2 estimated as 251.3
```

```
prevNN=forecast(fit,h=36)
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(h$mean,col=2)
lines(hd$mean,col=3)
lines(prev$mean,col=4)
lines(prevNN$mean,col=5)
legend('topleft',col=1:5,lty=1,legend=c('true','forecast with HW','forecast with damped HW','SARIMA','NN
```




```
print(sqrt(mean((prevNN$mean-sanfran_test)^2)))
```

```
## [1] 23.10702
```

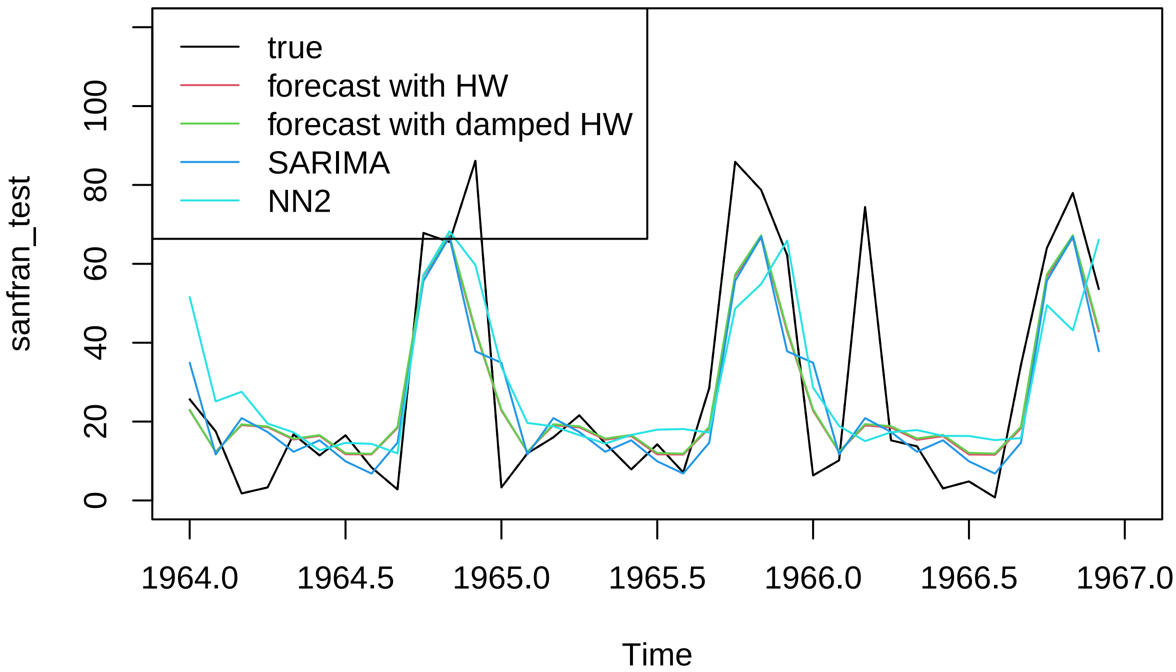
Forecasting are less efficient than with HW or SARIMA models.

We can try to tune these parameter manually, and select interesting order. For instance, we can choose this model … by cheating a little ;-) (*I use here the test set to select these parameter*)

```
fit=nnetar(sanfran_train,p=2,P=3,T=12,size=9)
print(fit)
```

```
## Series: sanfran_train
## Model:  NNAR(2,3,9)[12]
## Call:   nnetar(y = sanfran_train, p = 2, P = 3, size = 9, T = 12)
##
## Average of 20 networks, each of which is
## a 5-9-1 network with 64 weights
## options were - linear output units
##
## sigma^2 estimated as 118.1
```

```
prevNN2=forecast(fit,h=36)
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(h$mean,col=2)
lines(hd$mean,col=3)
lines(prev$mean,col=4)
lines(prevNN2$mean,col=5)
legend('topleft',col=1:5,lty=1,legend=c('true','forecast with HW','forecast with damped HW','SARIMA','NN
```




```
print(sqrt(mean((prevNN2$mean-sanfran_test)^2)))
```

```
## [1] 18.63158
```

Random Forest

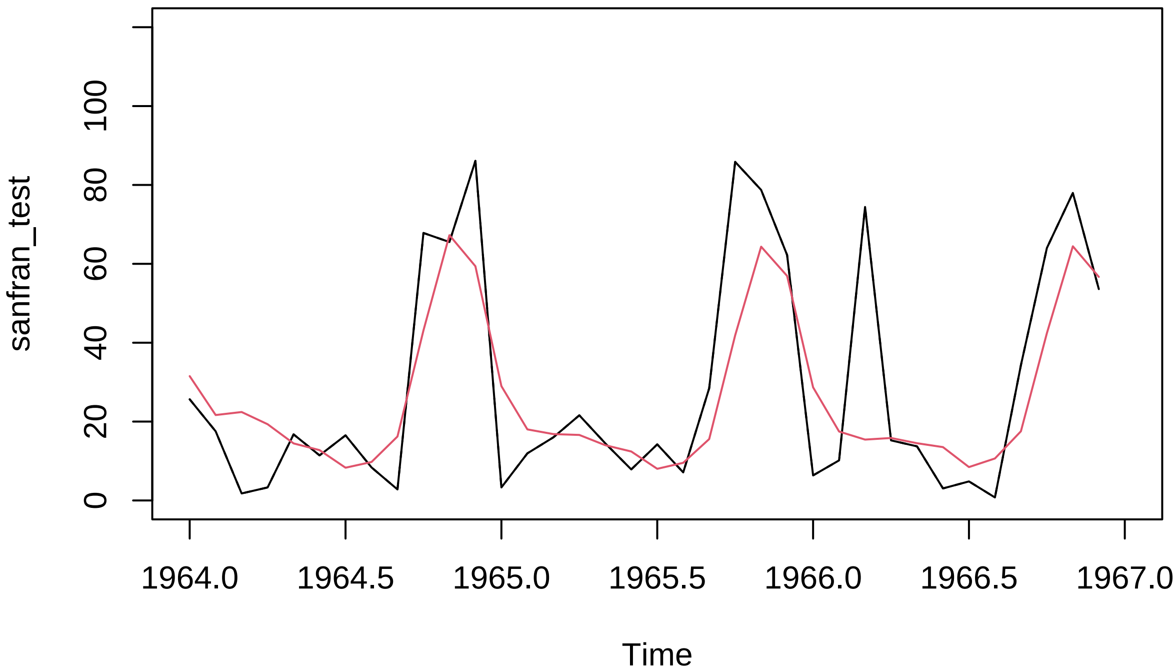
```
data=as.vector(sanfran_train)[1:13]
for (i in 1:(length(as.vector(sanfran_train))-13)){
data=rbind(data,as.vector(sanfran_train)[(i+1):(i+13)])
}
#data=matrix(as.vector(sanfran_train),nrow=29,ncol=13,byrow = TRUE)

library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

```
fitRF=randomForest(x=data[,-13], y=data[,13])
pred=rep(NULL,36)
newdata=tail(sanfran_train,12)
for (t in 1:36){
  pred[t]=predict(fitRF,newdata=newdata)
  newdata=c(newdata[-1],pred[t])
}
prevRF=ts(pred,start=c(1964,1),end=c(1966,12),frequency = 12)
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(prevRF,col=2)
```

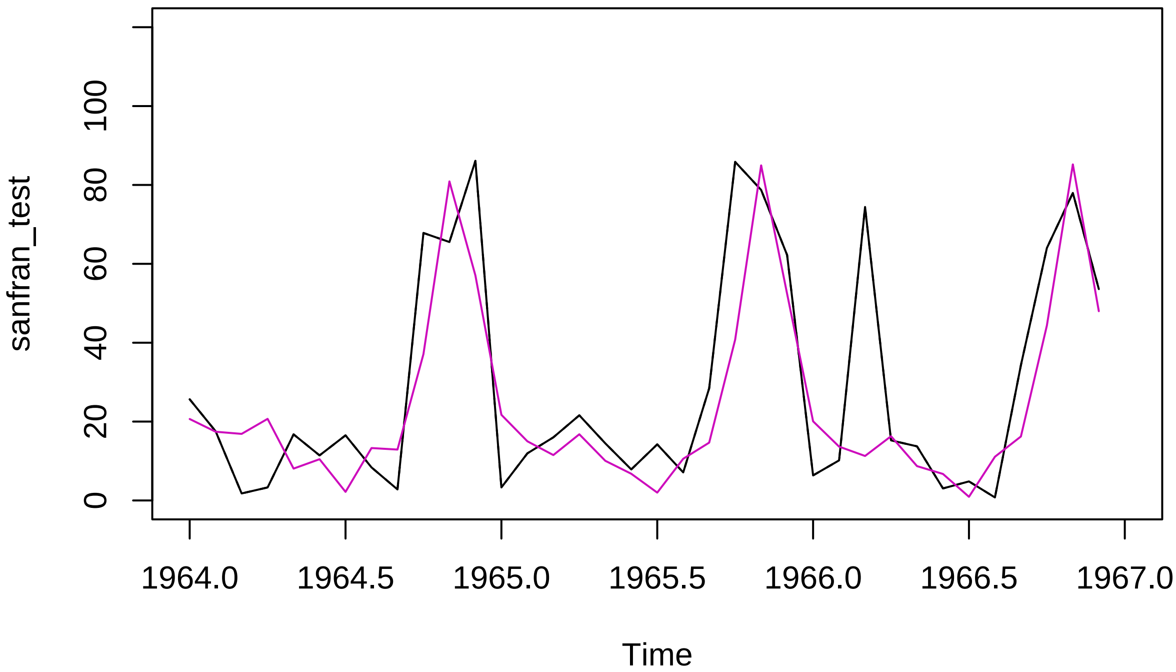



```
print(sqrt(mean((prevRF-sanfran_test)^2)))
```

```
## [1] 17.18772
```

SVM

```
library(e1071)
fitSVM=svm(x=data[,-13], y=data[,13])
pred=rep(NULL,36)
newdata=tail(sanfran_train,12)
for (t in 1:36){
  pred[t]=predict(fitSVM,newdata=matrix(newdata,1,12))
  newdata=c(newdata[-1],pred[t])
}
prevSVM=ts(pred,start=c(1964,1),end=c(1966,12),frequency = 12)
plot(sanfran_test,xlim=c(1964,1967),ylim=c(0,120))
lines(sanfran_test,lty=2)
lines(prevSVM,col=6)
```




```
print(sqrt(mean((prevSVM-sanfran_test)^2)))
```

```
## [1] 17.53055
```