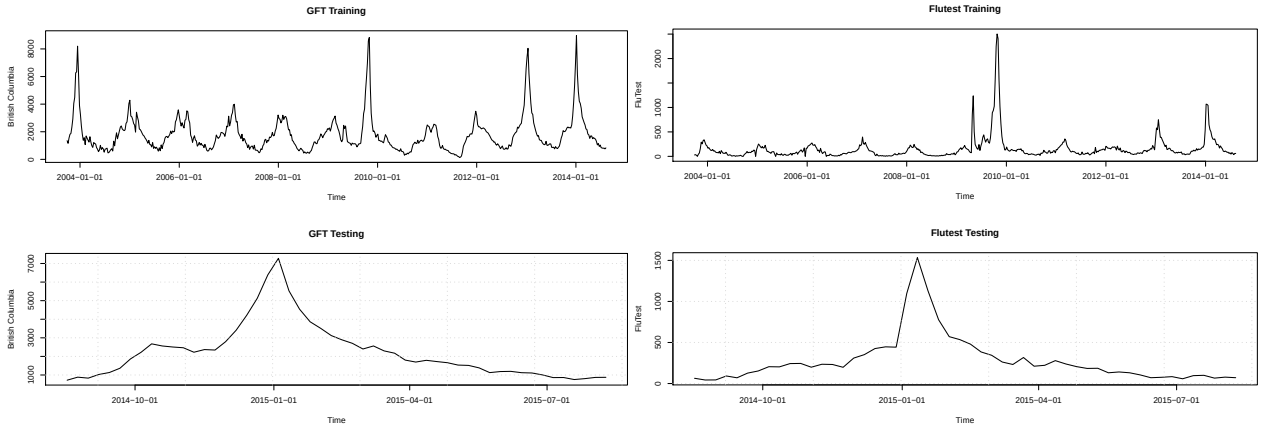
STA457_Assignment

Tuoyue Huang

2020/3/15

Data split and Modeling Flu Tests using Google Flue Trends

- Split the data set into two groups: training sample and test sample



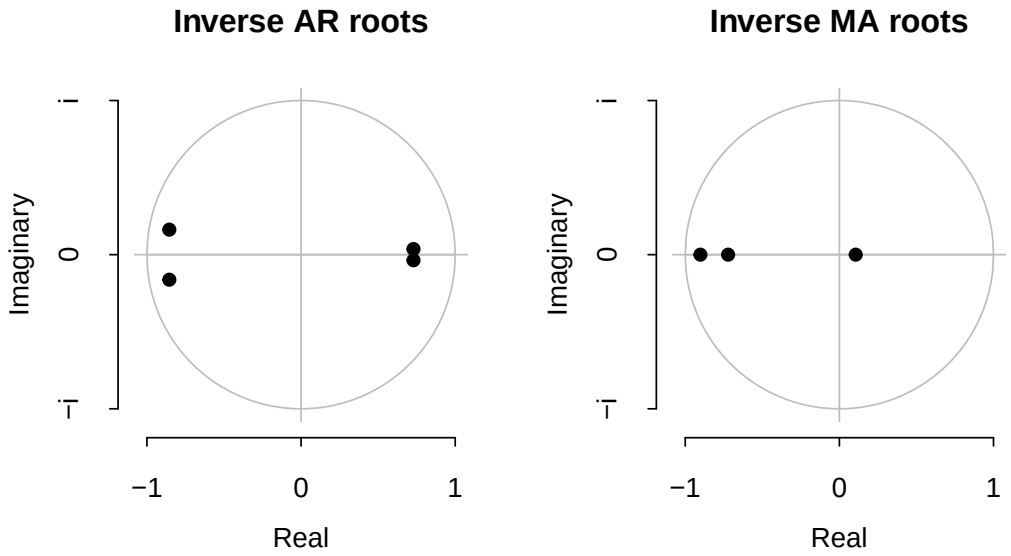
- Modeling Google Flu Trends

-We use the auto.arima function in forecast package to fit GFT, and as shown below.

```
##          ar1          ar2          ar3          ar4          ma1
##   -0.2508187    1.2035315    0.1925643   -0.4037494    1.5177007
##          ma2          ma3    intercept
##    0.4790093   -0.0689342 1793.0878754
```

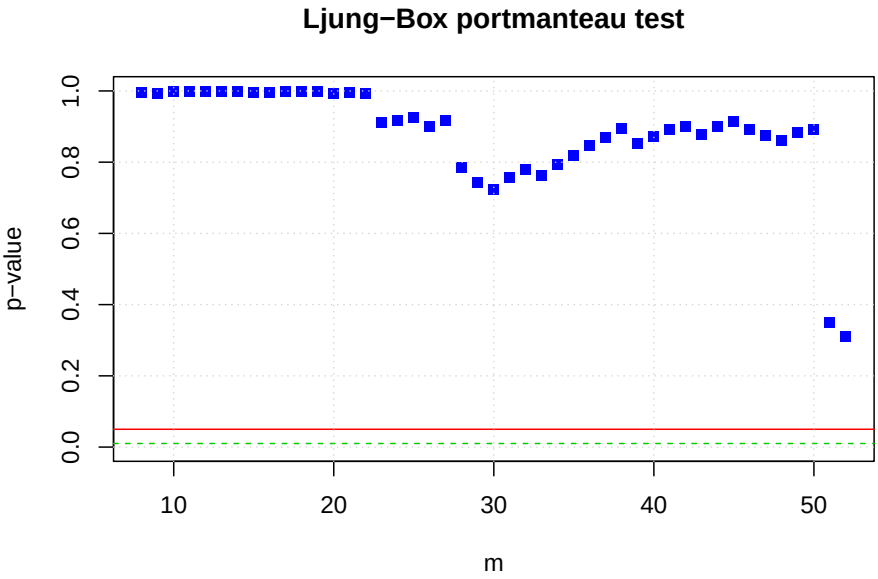
An ARMA(4, 3) model is selected based on AIC.

-Stability of the fitted ARMA model



The inverse AR and MA roots are all inside the unit circle so the fitted ARMA model is causal and invertible.

-Model adequacy



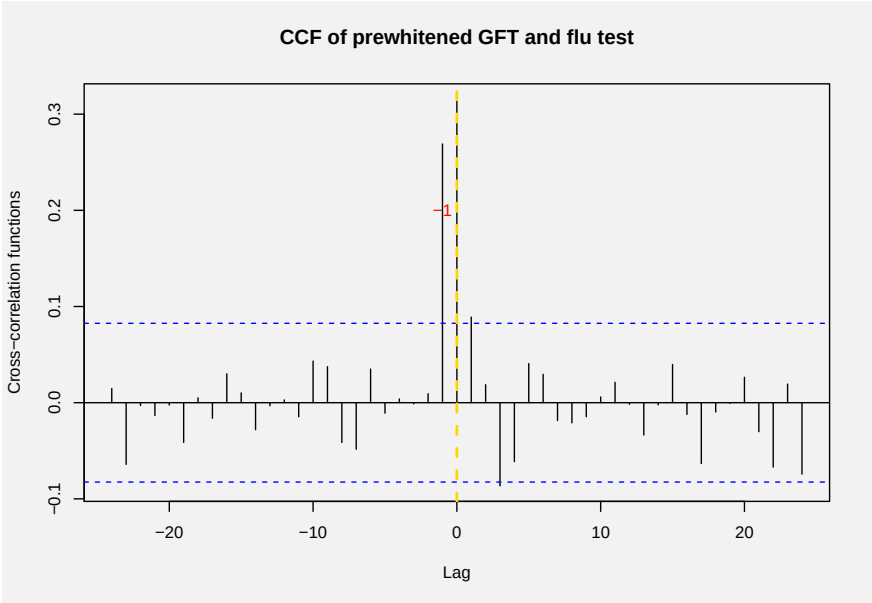
As shown above, the p-values of the test statistics are well above the 5% significance level so the fitted model is adequacy.

A. Transfer function noise modeling

1) Write the mathematical equation of the fitted transfer function noise model—using Google Flu Trends as the explanatory variables to predict Flu Tests.

- Modeling Flu Tests using Google Flue Trends

1.1)We use the CCF plot of Google Flue Trends to select the lag for tfn Model



As shown above, we will include GFT_t, GFT_{t-1} in our transfer function noise model. Although it is also significant at lag = 1, the CCF just touches the blue line and it is not reasonable for us to build a model based on future values of Google flue trends.

1.2) Transfer function noise model estimation with auto.arima function

	Est coef
ar1	1.67
ar2	-0.68
ma1	-0.48
ma2	-0.34
intercept	-95.28
GFT	0.08
GFT1	0.06

Therefore, our TFN model is the following:

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + v_0 GFT_t + v_1 GFT_{t-1} + a_t + \theta_1 a_{t-1} + \theta_2 a_{t-2}$$

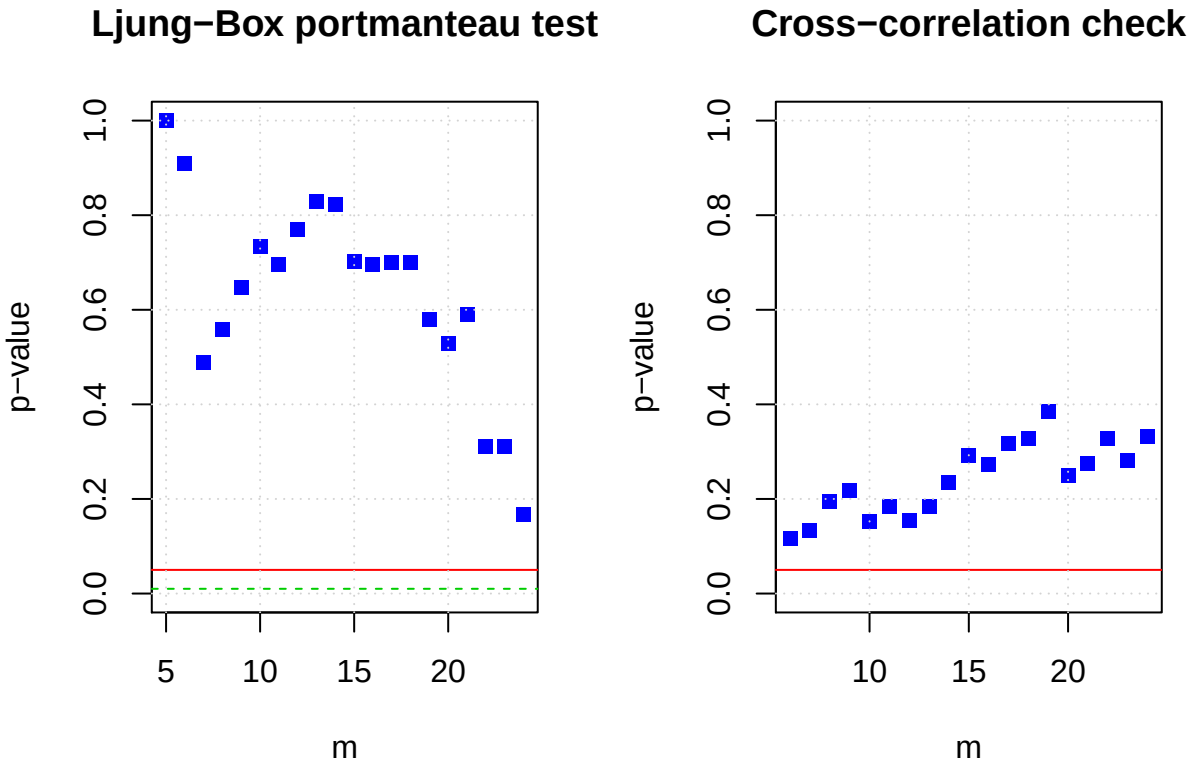
$$y_t = -95.28 + 1.67 y_{t-1} - 0.68 y_{t-2} + 0.08 GFT_t + 0.06 GFT_{t-1} + a_t - 0.48 a_{t-1} - 0.34 a_{t-2}$$

$$(1 - 1.67B + 0.68B^2) y_t = -95.28 + (0.08 + 0.06B) GFT_t + (1 - 0.48B - 0.34B^2) a_t$$

$$y_t = \frac{-95.28}{1 - 1.67B + 0.68B^2} + \frac{0.08 + 0.06B}{1 - 1.67B + 0.68B^2} GFT_t + \frac{1 - 0.48B - 0.34B^2}{1 - 1.67B + 0.68B^2} a_t$$

where y_t is the numbers of Flu Tests at time t, the first term is the intercept, GFT_t is the Google Flu Trends at time t as the explanatory variable and the last term is the error term, where a_t is a white noise process.

2) Is the fitted model adequate? Explain your answer.



First, we have the Ljung-Box test which is also the auto-correlation check to check whether the noise model is adequate, in other words, whether a_t is a white noise process.

Thus, the Null hypothesis is the residual of the TFN model(a_t) is a white noise process.

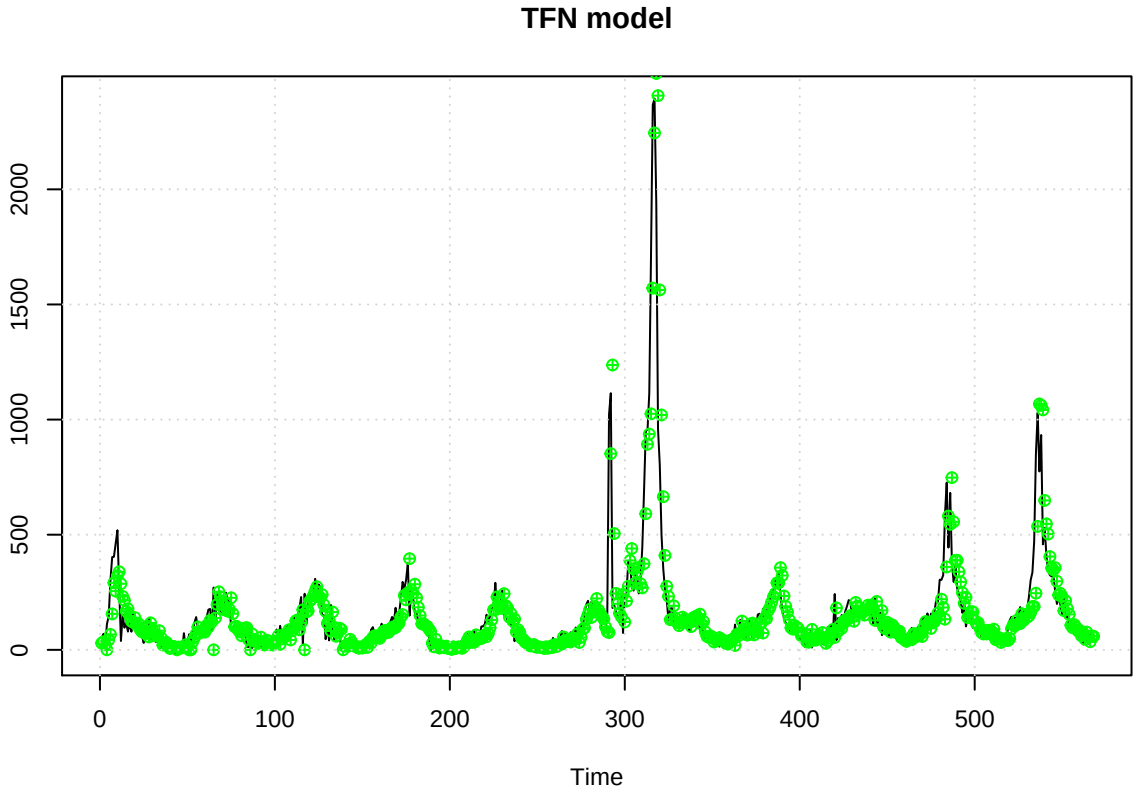
As shown above, the p-values of the test statistics are all above the 5%(the red line) significance level so we fail to reject H0 and conclude that the noise model is adequacy.

Further, we need the Cross-correlation check to test whether the noise a_t and the input series GFT_t are uncorrelated since it is an important assumption of the TFN model.

Thus, the Null hypothesis is the the noise a_t and the input series GFT_t are uncorrelated.

As shown above, the p-values of the test statistics are all above the 5%(the red line) significance level so we fail to reject H0 and conclude that a_t and GFT_t are uncorrelated.

-In addition, we could also look at the fitted TFN model to determine the adequacy



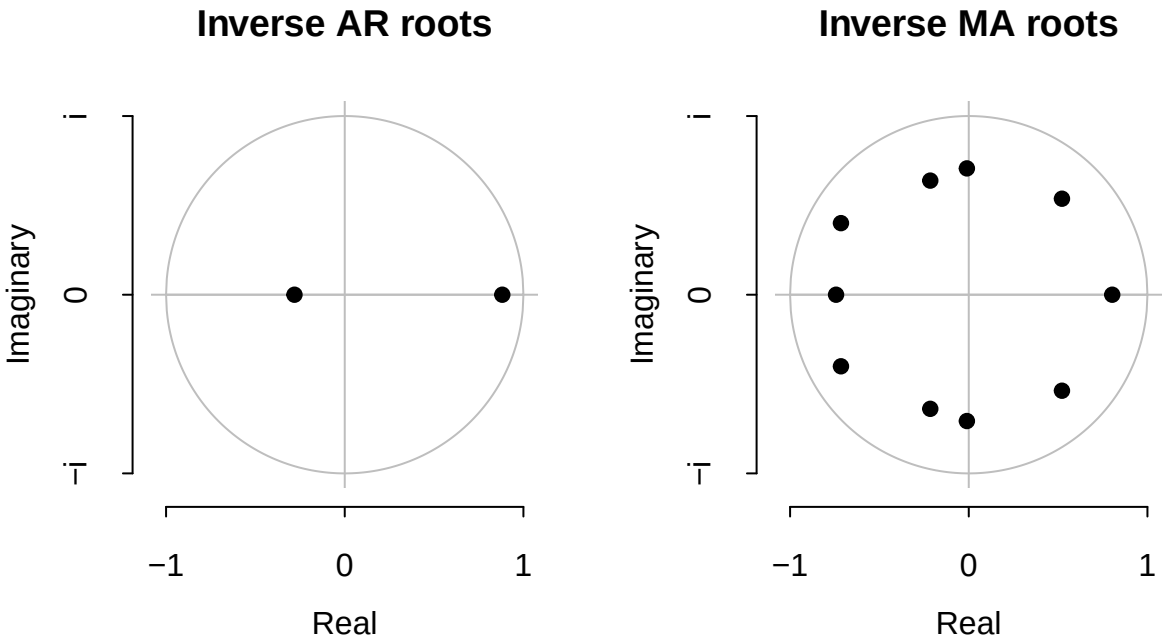
It basically captures the trend of the data, together with the above results of Ljung-Box test and Cross-correlation check, we could say the fitted TFN model is adequate.

B. Forecast evaluation – Consider ARIMA, TFN, NN, and NNX models

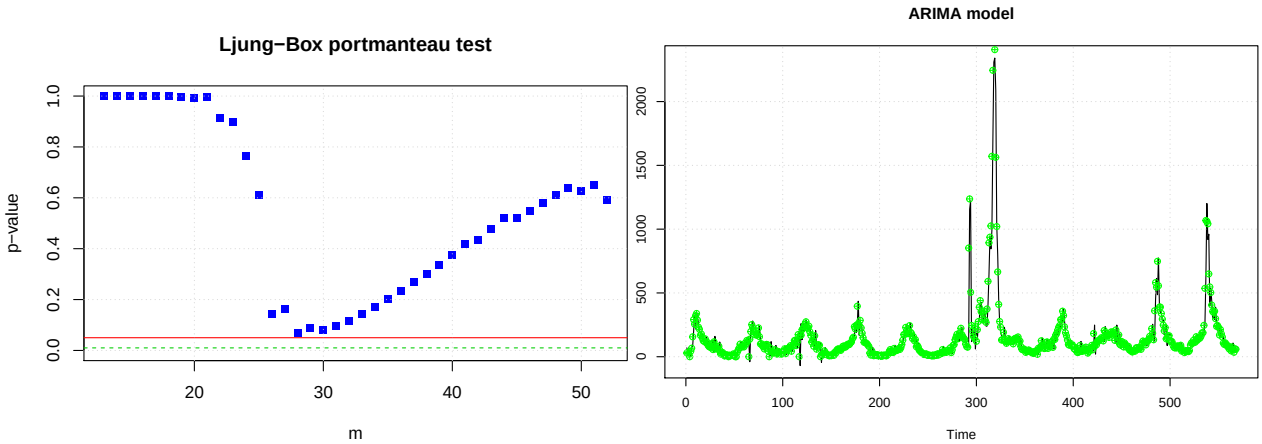
• First, we fit ARIMA, NN and NNX models

-ARIMA model:

```
##          ar1          ar2          ma1          ma2          ma3
##   0.60046401   0.24800431   0.77847438   0.23156072   0.02824909
##          ma4          ma5          ma6          ma7          ma8
##  -0.07236271  -0.07909194   0.01864754  -0.08921565  -0.11577841
##          ma9         ma10    intercept
##  -0.06937727  -0.05104630 148.66926785
```



By checking the ACF and PACF plots of training data set of flu test, we choose to fit an ARIMA(p=2,0,q=10). The inverse AR and MA roots are all inside the unit circle so the fitted ARIMA model is causal and invertible.

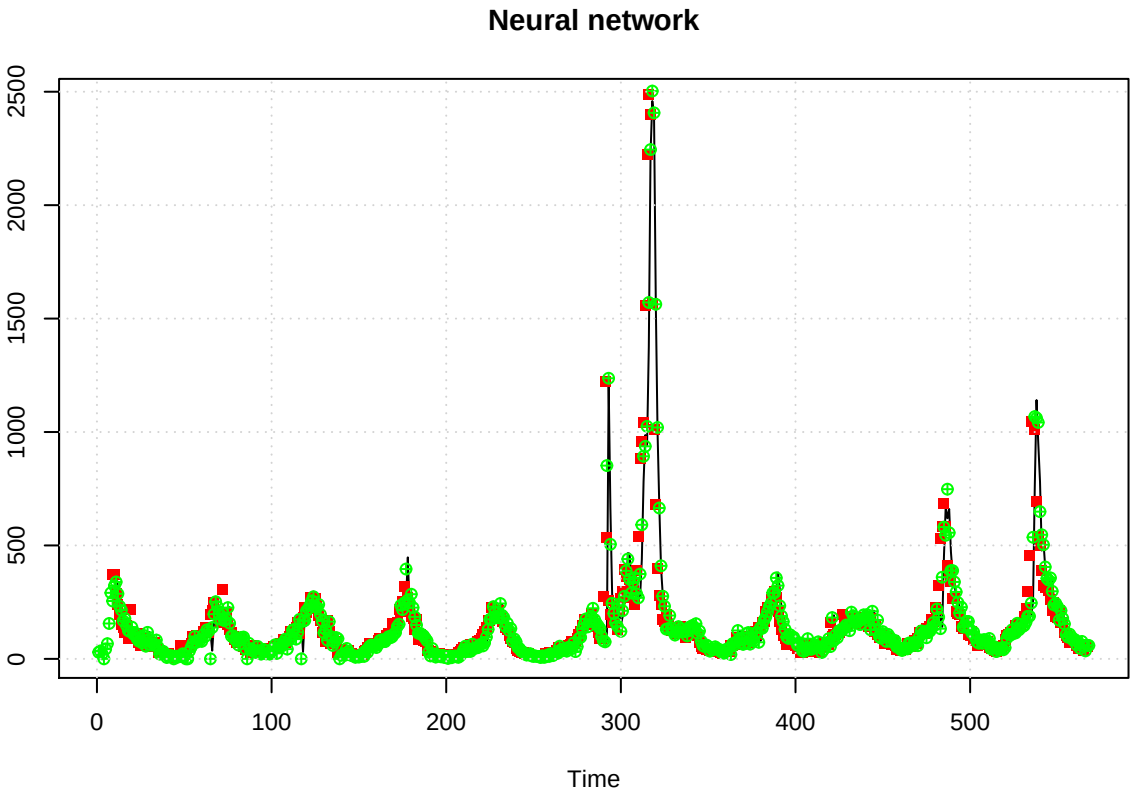


Similarly, we could use Ljung Box-test to check whether the noise model is adequate and also take a look at the fitted ARIMA model. As shown above, the p-values of the test statistics at different lags are all above the 5%(the red line) significance level so we fail to reject H0 and conclude that the noise model is adequacy. The fitted plot indicates the ARIMA model well capture the trend of the data.

-Neural Networks model: We fit Neural Network Time Series model through the nnetar function

```
## Series: fluTest.obs
## Model:  NNAR(8,4)
## Call:   forecast::nnetar(y = fluTest.obs)
##
## Average of 20 networks, each of which is
## a 8-4-1 network with 41 weights
## options were - linear output units
##
## sigma^2 estimated as 2792
```

```
## Series: data.obs[, 1]
## Model:  NNAR(8,6)
## Call:   forecast::nnetar(y = data.obs[, 1], xreg = data.obs[, -1])
##
## Average of 20 networks, each of which is
## a 10-6-1 network with 73 weights
## options were - linear output units
##
## sigma^2 estimated as 1393
```

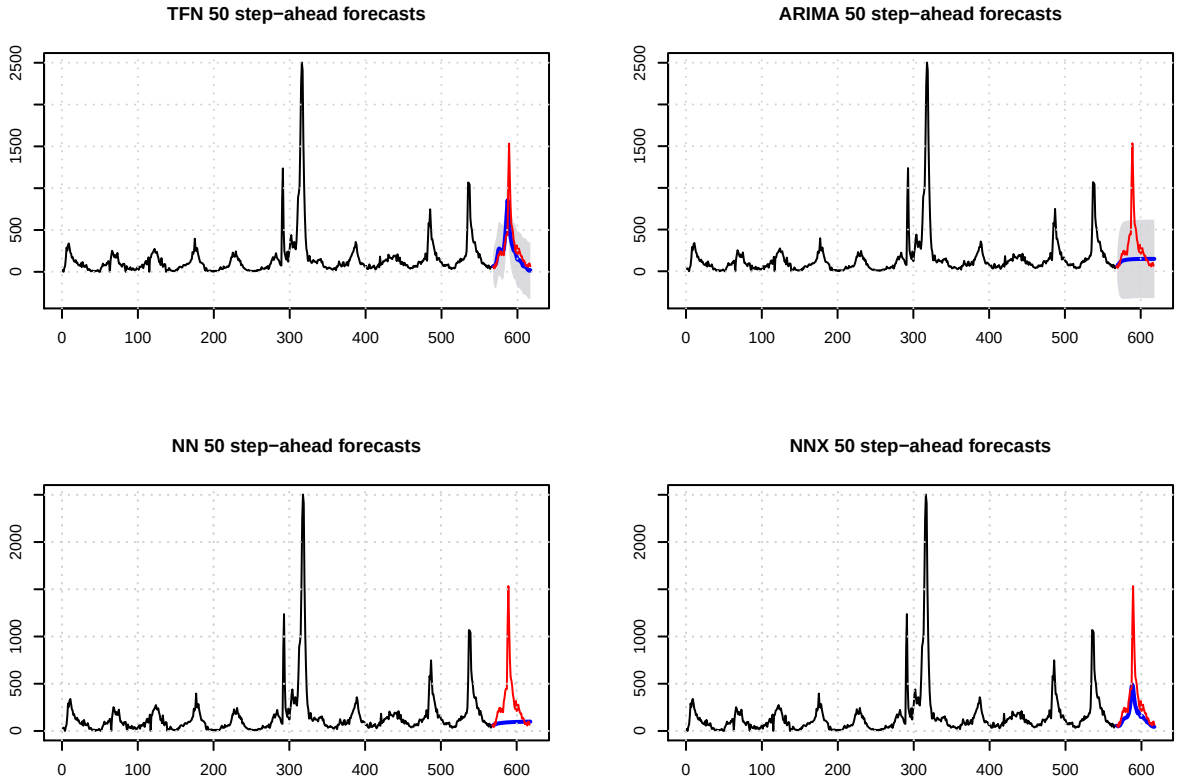


We define NN model is the NNAR(8,4) and NNX model is the NNAR(8,6), where it includes the two Google Flu Trends variables. By the fitted plot, we can see the data are well predicted by the two models.

The following plots show the forecast of the flu test in the next 50 weeks with our fitted TFN, ARIMA, NN, NNX models. The red line is the actual data, where the blue line is the predicting results.

From the plots, we cay say the TFN model and NNX generally capture the fluctuating trend of the data during the time period, while the TFN model is doing slightly better than the NNX. However, the ARIMA and NN model only get the short term changes and become flat in the long term.

We still need further forecast statistics through accuracy function to evaluate the forecast performance of these models.



1) Calculate the forecast performance in the training sample, and answer which model performs best in each forecast measure.

Table 2: Model forecasting performance(Training sample)

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
1.ARIMA	0.14	81.20	38.41	-Inf	Inf	1.00	0.00
2.TFN	0.24	72.30	36.63	NaN	Inf	0.95	0.00
3.NN	-0.25	52.84	27.05	-Inf	Inf	0.71	-0.02
4.NNX	0.02	37.32	20.41	-Inf	Inf	0.53	-0.05

```
##            ME RMSE MAE MASE ACF1
## Best model  4    4   4    4    1
```

A good performance of a model means it has lower absolute value of forecast statistics like ME,RMSE and etc. For instance, we look at the ME column, NNX model gives the lowest absolute value of ME, so it's the best one among them. We ignore the ones with NaN and Inf. In the training sample, we can see NNX model performs best on ME, RMSE, MAE and MASE, while ARIMA model wins at ACF1.

2)Calculate the forecast performance in the test sample for the following lead time:h= 1;4;8;50 and answer which model performs best in each forecast measure.

-Short-run accuracy (h=1)

Table 3: Model forecasting performance(h=1)

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
1.ARIMA	-17.74	17.74	17.74	-41.25	41.25	0.46	NA
2.TFN	-25.88	25.88	25.88	-60.19	60.19	0.67	NA
3.NN	-18.68	18.68	18.68	-43.44	43.44	0.49	NA
4.NNX	-14.04	14.04	14.04	-32.66	32.66	0.37	NA

```
##            ME RMSE MAE MPE MAPE MASE
## Best model  4    4   4   1    1    4
```

At h=1, which is forecasting of one week in the test sample, we ignore the ACF1 column as it produce NA. Analyze similarly as the training sample, among the remaining ones, we can see NNX model performs best on ME, RMSE, MAE and MASE, while ARIMA model wins at MPE and MAPE.

-Short-run accuracy (h=4)

Table 4: Model forecasting performance(h=4)

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
1.ARIMA	-18.39	23.95	21.31	-38.24	41.42	0.56	NA
2.TFN	-10.87	21.40	21.11	-27.70	38.84	0.55	NA
3.NN	-4.24	19.73	17.29	-17.31	31.50	0.45	NA
4.NNX	7.94	20.77	17.11	4.19	25.40	0.45	NA

```
##            ME RMSE MAE MPE MAPE MASE
## Best model  3    3   4   4    4    3
```

At h=4, which is forecasting of four weeks in the test sample, we ignore the ACF1 column as it produce NA. Among the remaining ones, we can see NNX model performs best on MAE, MPE and MAPE, while NN model wins at ME, RMSE and MASE. This indicates Neural Network model is good at short run predicting.

-Short-run accuracy (h=8)

Table 5: Model forecasting performance(h=8)

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
1.ARIMA	17.79	46.04	37.64	-4.48	35.35	0.98	NA
2.TFN	-11.32	27.37	21.67	-16.03	25.71	0.56	NA
3.NN	44.98	71.69	55.75	17.89	42.29	1.45	NA
4.NNX	47.64	64.82	52.23	27.51	38.12	1.36	NA

```
##            ME RMSE MAE MPE MAPE MASE
## Best model  2    2   2   1    2    2
```

At h=8, which is forecasting of eight weeks in the test sample, we ignore the ACF1 column as it produce NA. Among the remaining ones, we can see TFN model performs best on ME, RMSE, MAE, MAPE and MASE, while ARIMA model wins at MPE. This indicates TFN model is good at medium run predicting. We may expect to see it perform well when h=50 in the long run.

-Long-run accuracy (h=50)

Table 6: Model forecasting performance(h=50)

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
1.ARIMA	154.25	325.12	183.19	14.45	53.97	4.78	NA
2.TFN	62.42	162.04	90.30	17.74	31.14	2.35	NA
3.NN	201.81	352.75	210.00	41.88	55.20	5.48	NA
4.NNX	133.30	229.96	134.33	35.42	37.52	3.50	NA

```
##             ME RMSE MAE MPE MAPE MASE
## Best model  2    2   2   1    2    2
```

At h=50, which is forecasting of fifty weeks in the test sample, we ignore the ACF1 column as it produce NA. Among the remaining ones, we can see TFN model performs best on ME, RMSE, MAE, MAPE and MASE, while ARIMA model wins at MPE. This indicates TFN model is also good at long run predicting as expected. This also matches the result of plot of 50-step-forecasting.

Appendix and Cite

```
Prof.JEN-WEN LIN "Time-Series-Forecasting-Examples.html"

knitr::opts_chunk$set(echo = TRUE)
library(knitr)
library(readxl)
library(MASS)
library(forecast)
library(timeSeries)
library(timeDate)
library(sarima)
library(dynlm)
library(data.table)

# PreWhiten.ar
# Prewhitening time series x based on a fitted AR model
# x: Time series to be transformed (n x 1 vector)
# ar: Autoregressive coefficient estimates (p-1 x 1 vector)
PreWhiten.ar<- function(x , ar = NULL){
  if(is.null(ar)) print(" autoregressive coefficients are empty!")
  pwData = numeric(0)
  pwData = filter(x, c(1, -ar),method=c("convo"),sides=1)
  pwData[!is.na(pwData)]
}

# PreWhiten.arma
# Prewhitening time series x based on a fitted ARMA model
# x: Time series to be transformed (n x 1 vector)
# ar: Autoregressive coefficient estimates (p x 1 vector)
# ma: Moving-average coefficient estimates (q x 1 vector)
PreWhiten.arma<- function(x , ar = NULL, ma = 0){
  if(is.null(ar) && is.null(ma)) print("both ar and ma coefficients are empty!")
  pwData = numeric(0)
  m = as(modelCoef(new("ArmaModel", ar = ar, ma = ma)), "list")
```

```
  eps = numeric(length(x))
  pwData = xarmaFilter(m, x =x, eps = eps, whiten = TRUE)
  pwData[!is.na(pwData)]
}

# LBTest
# Return thet p-values of Ljung-Box portmanteau tests for a fitted ARMA model
# res: residuals from a fitted ARMA model  (n x 1 vector)
# nPQ: # of model parameters (integer)
# m: test inputs (integer)
LBTest<- function(res, nPQ = 0, m = 24, ifPlot = FALSE){
  stopifnot(nPQ >= 0, m >= 1, m > nPQ)
  n <- length(res)
  lags <- 1:m
  df <- (nPQ+1):m
  ra <- (acf(res, lag.max = m, plot = FALSE)$acf)[-1]
  QQ <- n * (n + 2) * cumsum((ra^2)/(n - (1:m)))[df]

  pv <- 1 - pchisq(QQ, df)
  QQ <- round(QQ, 2)
  a <- matrix(c(df, QQ, pv), ncol = 3)
  dimnames(a) <- list(rep("", length(QQ)), c("m", "Qm", "pvalue"))
  if(ifPlot){
    plot(x = a[,1],y = a[,3],
         ylim = c(0,1), pch = 15, col =4,
         ylab = "p-value", xlab = "m",
         main = "Ljung-Box portmanteau test")
    abline(h =0.05, col =2)
    abline(h =0.01, col =3, lty = 2)
    grid()
  }else {
    a
  }
}

GFluTrends<-read_excel("/Users/huangtuoyue/Downloads/case_study_1_fluwatch.xlsx",
                       sheet="Google Flu Trends", skip = 1)
fluWatch<-read_excel("/Users/huangtuoyue/Downloads/case_study_1_fluwatch.xlsx",
                     sheet="FluWatch-BC", skip = 2)
tim<-timeSequence(from = "2003-09-28", to = "2015-08-09", by = "week")
tim1<-timeSequence(from = "2003-09-07", to = "2015-08-23", by = "week")
GFT<- timeSeries(GFluTrends[,"British Columbia"], charvec = tim)
fluTest<- timeSeries(fluWatch[,"FluTest"], charvec = tim1)
# Training and testing
GFT.obs = window(GFT, start = "2003-09-28", end = "2014-08-10")
GFT.test = window(GFT, start = "2014-08-17", end = "2015-08-09")
par(mfrow = c(2,1), cex = 0.5)
plot(GFT.obs, main = "GFT Training")
plot(GFT.test, main = "GFT Testing")
grid()

fluTest.obs = window(fluTest, start = "2003-09-28", end = "2014-08-10")
fluTest.test = window(fluTest, start = "2014-08-17", end = "2015-08-09")
```

```r
par(mfrow = c(2,1), cex = 0.5)
plot(fluTest.obs, main = "Flutest Training")
plot(fluTest.test, main = "Flutest Testing")
grid()
#2.1 Model selection and estimation
#We use the auto.arima function in forecast package to fit GFT, and as shown below.
mod.arma<-auto.arima(GFT.obs, max.p = 52, max.q = 52, stationary = TRUE)
p = mod.arma$arma[1]; q = mod.arma$arma[2]
coef(mod.arma)
plot(mod.arma)
npq = sum(mod.arma$arma[c(1,2)])
LBTest(mod.arma$residuals, nPQ = npq, m = 52, ifPlot = TRUE)
mod = mod.arma;nAR = mod$arma[1]; nMA = mod$arma[2]

if(nMA!=0){
  xf = PreWhiten.arma(GFT.obs, ar = mod$coef[1:nAR],
                      ma = mod$coef[(1:nMA)+nAR])[-(1:nAR)]
  yf = PreWhiten.arma(fluTest.obs, ar = mod$coef[1:nAR],
                      ma=mod$coef[(1:nMA)+nAR])[-(1:nAR)]
}else{
  xf = PreWhiten.arma(GFT.obs, ar = mod$coef[1:nAR],
                      ma = 0)[-(1:nAR)]
  yf = PreWhiten.arma(fluTest.obs, ar = mod$coef[1:nAR],
                      ma=0)[-(1:nAR)]
}

par(cex=0.75,bg="gray95")
ccf(c(xf), c(yf), lwd=1, ylab="Cross-correlation functions",
    main="CCF of prewhitened GFT and flu test")
abline(v=0, col="gold", lwd=2, lty="dashed")
text(-1, 0.2, "-1", col=2)
y<-fluTest
x<-GFT
dat<- cbind(y,x, lag(x))[-c(1:4,625),]
colnames(dat)<-c("fluTest", "GFT", "GFT1")
data<- timeSeries(dat, charvec = tim)

data.obs = window(data, start = "2003-10-05", end = "2014-08-10")
data.test = window(data, start = "2014-08-17", end = "2015-07-26") #zhuyi

mod.tfn = auto.arima(data.obs[,1], xreg = data.obs[,-1], stationary = TRUE)
knitr::kable(coef(mod.tfn), digits = 2,caption = "Coefficient of TFN Model",
             col.names=(c("Est coef")), format = "markdown")
m = 24
lags = 1:m
df <- (2+3+1):m
n = length(mod.tfn$res)
rccf = ccf(mod$residuals,mod.tfn$residuals, plot = FALSE, lag.max = m)$acf[-(1:m)]
Qm = n* (n + 2) * cumsum((rccf^2)/(n - (0:m)))[df]
pv <- 1 - pchisq(Qm, df)
a = cbind(df, Qm,pv)

par(mfrow = c(1,2))
```

```r
LBTest(mod.tfn$res, nPQ = 4, ifPlot = TRUE)
plot(x = a[,1],y = a[,3],
     ylim = c(0,1), pch = 15, col =4,
     ylab = "p-value", xlab = "m",
     main = "Cross-correlation check")
abline(h =0.05, col =2)
grid()
par(mfrow = c(1,1), cex = 0.75)
ts.plot(mod.tfn$fitted, ylab = "", main ="TFN model")
lines(c(fluTest.obs), pch = 10, col = "green", type ="p")
grid()
#acf(fluTest.obs)
#pacf(fluTest.obs)
mod.arima <- arima(fluTest.obs, order = c(2,0,10))
#mod.arima <- auto.arima(fluTest.obs)
coef(mod.arima)
plot(mod.arima)
npq_arima = sum(mod.arima$arma[c(1,2)])
LBTest(mod.arima$residuals, nPQ = npq_arima, m = 52, ifPlot = TRUE)

par(mfrow = c(1,1), cex = 0.75)
ts.plot(forecast(mod.arima)$fitted , ylab = "", main ="ARIMA model")
lines(c(fluTest.obs), pch = 10, col = "green", type ="p")
grid()
set.seed(10000)
mod.nn = forecast::nnetar(fluTest.obs)
mod.nn

mod.nnx = forecast::nnetar(data.obs[,1], xreg = data.obs[,-1])
mod.nnx

par(mfrow = c(1,1), cex = 0.75)
ts.plot(mod.nn$fitted, ylab = "", main ="Neural network")
lines(c(mod.nnx$fitted), col = "red", lwd = 2, type ="p", pch = 15)
lines(c(fluTest.obs), pch = 10, col = "green", type ="p")
grid()
par(mfrow=c(2,2),cex=0.5)
plot(forecast(mod.tfn, xreg = data.test[,-1], level = 95),
     main = "TFN 50 step-ahead forecasts")
lines(c(rep(NA,567),fluTest.test[-c(51,52)]),
      col = "red",pch=18); grid()
plot(forecast(mod.arima,h = 50, level = 95),
     main = "ARIMA 50 step-ahead forecasts")
lines(c(rep(NA,567), fluTest.test[-c(51,52)]),
      col = "red",pch =18); grid()
plot(forecast(mod.nn, h=50, level = 95),
     main = "NN 50 step-ahead forecasts"); grid()
lines(c(rep(NA,567), fluTest.test[-c(51,52)]),
      col = "red",pch =18); grid()
plot(forecast(mod.nnx, h=50, xreg = data.test[,-1], level = c(95)),
     main = "NNX 50 step-ahead forecasts"); grid()
lines(c(rep(NA,567), fluTest.test[-c(51,52)]),
      col = "red",pch =18); grid()
```

```
table.obs <- round(rbind(accuracy(mod.arima), accuracy(mod.tfn),
                         accuracy(mod.nn), accuracy(mod.nnx)),digits = 2)
rownames(table.obs) = c("1.ARIMA","2.TFN","3.NN","4.NNX")

knitr::kable(table.obs,
             caption = "Model forecasting performance(Training sample)")
perform0<-t(as.matrix(apply(abs(table.obs), 2, which.min)))
perform0 <- as.data.frame(perform0)
rownames(perform0)= "Best model"
perform0[,-c(4,5)]

h = 1
week1_arima <- forecast(mod.arima, h = h)
row1 <- accuracy(f=week1_arima, x=data.test[1:h])

week1_tfn <- forecast(mod.tfn, xreg = data.test[1:h,-1], h = h)
row2 <- accuracy(f=week1_tfn, x=data.test[1:h])

week1_nn <- forecast(mod.nn, h = h)
row3 <- accuracy(f=week1_nn, x=data.test[1:h])

week1_nnx <- forecast(mod.nnx, xreg = data.test[1:h,-1], h = h)
row4 <- accuracy(f=week1_nnx, x=data.test[1:h])

table.obs1 <- round(rbind(
  row1[2,], row2[2,], row3[2,], row4[2,]), digits = 2)
row.names(table.obs1) = c("1.ARIMA","2.TFN","3.NN","4.NNX")


knitr::kable(table.obs1,
             caption = "Model forecasting performance(h=1)")
perform1<-t(as.matrix(apply(abs(table.obs1), 2, which.min)))
perform1 <- as.data.frame(perform0)
rownames(perform1)= "Best model"
perform1[,-c(7)]
h = 4
week1_arima <- forecast(mod.arima, h = h)
row1 <- accuracy(f=week1_arima, x=data.test[1:h])

week1_tfn <- forecast(mod.tfn, xreg = data.test[1:h,-1], h = h)
row2 <- accuracy(f=week1_tfn, x=data.test[1:h])

week1_nn <- forecast(mod.nn, h = h)
row3 <- accuracy(f=week1_nn, x=data.test[1:h])

week1_nnx <- forecast(mod.nnx, xreg = data.test[1:h,-1], h = h)
row4 <- accuracy(f=week1_nnx, x=data.test[1:h])

table.obs2 <- round(rbind(
  row1[2,], row2[2,], row3[2,], row4[2,]), digits = 2)
row.names(table.obs2) = c("1.ARIMA","2.TFN","3.NN","4.NNX")

knitr::kable(table.obs2,
```

```
             caption = "Model forecasting performance(h=4)")
perform2<-t(as.matrix(apply(abs(table.obs2), 2, which.min)))
perform2 <- as.data.frame(perform2)
rownames(perform2)= "Best model"
perform2[,-c(7)]

h = 8
week1_arima <- forecast(mod.arima, h = h)
row1 <- accuracy(f=week1_arima, x=data.test[1:h])

week1_tfn <- forecast(mod.tfn, xreg = data.test[1:h,-1], h = h)
row2 <- accuracy(f=week1_tfn, x=data.test[1:h])

week1_nn <- forecast(mod.nn, h = h)
row3 <- accuracy(f=week1_nn, x=data.test[1:h])

week1_nnx <- forecast(mod.nnx, xreg = data.test[1:h,-1], h = h)
row4 <- accuracy(f=week1_nnx, x=data.test[1:h])

table.obs3 <- round(rbind(
  row1[2,], row2[2,], row3[2,], row4[2,]), digits = 2)
row.names(table.obs3) = c("1.ARIMA","2.TFN","3.NN","4.NNX")

knitr::kable(table.obs3,
             caption = "Model forecasting performance(h=8)")
perform3<-t(as.matrix(apply(abs(table.obs3), 2, which.min)))
perform3 <- as.data.frame(perform3)
rownames(perform3)= "Best model"
perform3[,-c(7)]
h = 50
week1_arima <- forecast(mod.arima, h = h)
row1 <- accuracy(f=week1_arima, x=data.test[1:h])

week1_tfn <- forecast(mod.tfn, xreg = data.test[1:h,-1], h = h)
row2 <- accuracy(f=week1_tfn, x=data.test[1:h])

week1_nn <- forecast(mod.nn, h = h)
row3 <- accuracy(f=week1_nn, x=data.test[1:h])

week1_nnx <- forecast(mod.nnx, xreg = data.test[1:h,-1], h = h)
row4 <- accuracy(f=week1_nnx, x=data.test[1:h])

table.obs4 <- round(rbind(
  row1[2,], row2[2,], row3[2,], row4[2,]), digits = 2)
row.names(table.obs4) = c("1.ARIMA","2.TFN","3.NN","4.NNX")

knitr::kable(table.obs4,
             caption = "Model forecasting performance(h=50)")
perform4<-t(as.matrix(apply(abs(table.obs4), 2, which.min)))
perform4 <- as.data.frame(perform4)
rownames(perform4)= "Best model"
perform4[,-c(7)]
```