Recap of Exploratory visualization: Bike example, daily data

Lim, Kyuson

October, 2021

Motivation/Idea

The motivation is to predict the optimal values of consumers (count of causal users) for data based on the temperature.

I assume for some data given in the class without statement in the paper that the number of users are casually influenced by the temperature of daily records.

To investigate this issue, I would compare ARIMA, TFN (Transfer noise function model) and lastly fit neural network model to predict for the daily count of causal users from date of `2012-12-25` to `2012-12-31`.

For the training dataset, the data of both temperature and causal variable is used from the date of `2011-02-01` to `2012-12-24`, as to account for the possible numbers of lagged time sequence in the TFN model for January data.

In more detail, I would fit for the causal user variable against temperature because my asumption is that the lower the temperature is the less the number of causal users are to come for the bikes. This causal relation could be modelled with TFN model.

Set up

Such time series modelling is based on the use of unit root test imported from reference list for stationarity of ARIMA/ARMA model to use with.

Also, the Ljung-Box test code is provided from University of Toronto STA457 class directly from the lecture note for testing the stationary of ARMA model by the p-value.

Fitting the data

```r
setwd("/Users/kyusoinlims/Desktop/STATS 744")
#d at<-ts(read.csv(file.choose()))
dat<-ts(read.csv('day.csv'))
# count of customers
Timeline<-seq(as.Date("2011-01-01"), as.Date("2012-12-31"), by='days')
del.dat =xts(dat[,14], frequency = 7,Timeline)
# temperature
temp.dat=xts(dat[,10], frequency = 7,Timeline)
```

Nobel prize in economics: Granger test

From the famous example of egg and chicken data, which one to cause the other, the Granger test could be used for the time series data to verify if there truely exists for the causal relationship between two data we are trying to test with.

```
# Granger test
cons_us<-diff(dat[,14])
temper_no<-diff(dat[,10])
# does temperature cause casual consumer?
grangertest(cons_us~temper_no)
```

```
## Granger causality test
##
## Model 1: cons_us ~ Lags(cons_us, 1:1) + Lags(temper_no, 1:1)
## Model 2: cons_us ~ Lags(cons_us, 1:1)
##   Res.Df Df      F    Pr(>F)
## 1    726
## 2    727 -1 12.278 0.0004865 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# or the other way?
grangertest(temper_no~cons_us)
```

```
## Granger causality test
##
## Model 1: temper_no ~ Lags(temper_no, 1:1) + Lags(cons_us, 1:1)
## Model 2: temper_no ~ Lags(temper_no, 1:1)
##   Res.Df Df      F Pr(>F)
## 1    726
## 2    727 -1 0.5971   0.44
```

As to verify from the first test for the difference of data in two datasets, we can find that the first test to be significant and conclude that the normalized temperature data actually comes first for the causal influence on the number of casual users in daily data.

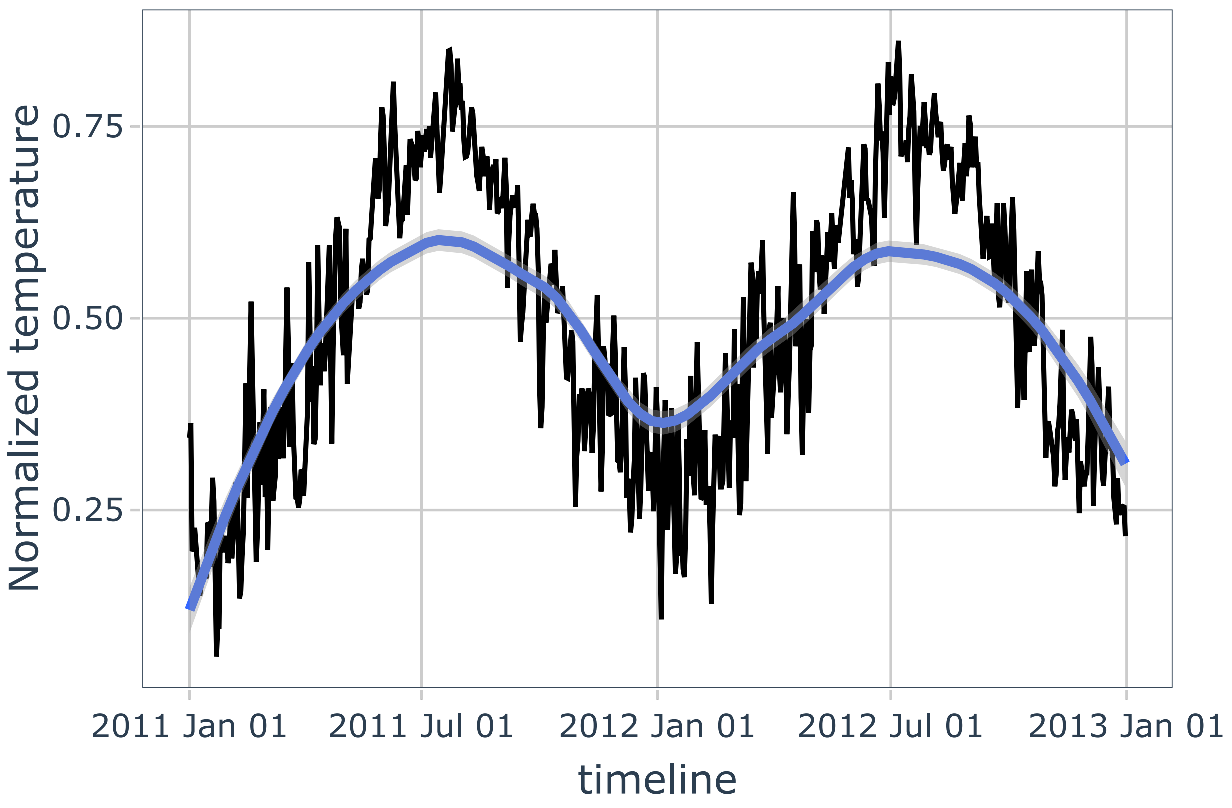
This validate for us to proceed for the TFN model where the casual relationship could be eliminated by the prewhitening process to model the number of causal users against the other dataset, normalized temperature.

Use the data to study the overall trend of two datasets

```
#split training and forecasting sample
temp.obs = window(temp.dat, start = "2011-02-01", end = "2012-12-24")
temp.test = window(temp.dat, start = "2012-12-25", end = "2012-12-31")
tm.obs = window(del.dat, start =  "2011-02-01", end = "2012-12-24")
tm.test = window(del.dat, start = "2012-12-25", end = "2012-12-31")
# temperature
# trend smoothed line
df<-data.frame(Temperature = dat[,10], Timeline)
colnames(df)<-c('Temperature')
plot_1<-ggplot(df, aes(Timeline, Temperature)) + geom_line(na.rm=TRUE)+ stat_smooth()+ theme_tq()+ xlab
# interactive plot
ggplotly(plot_1)
```

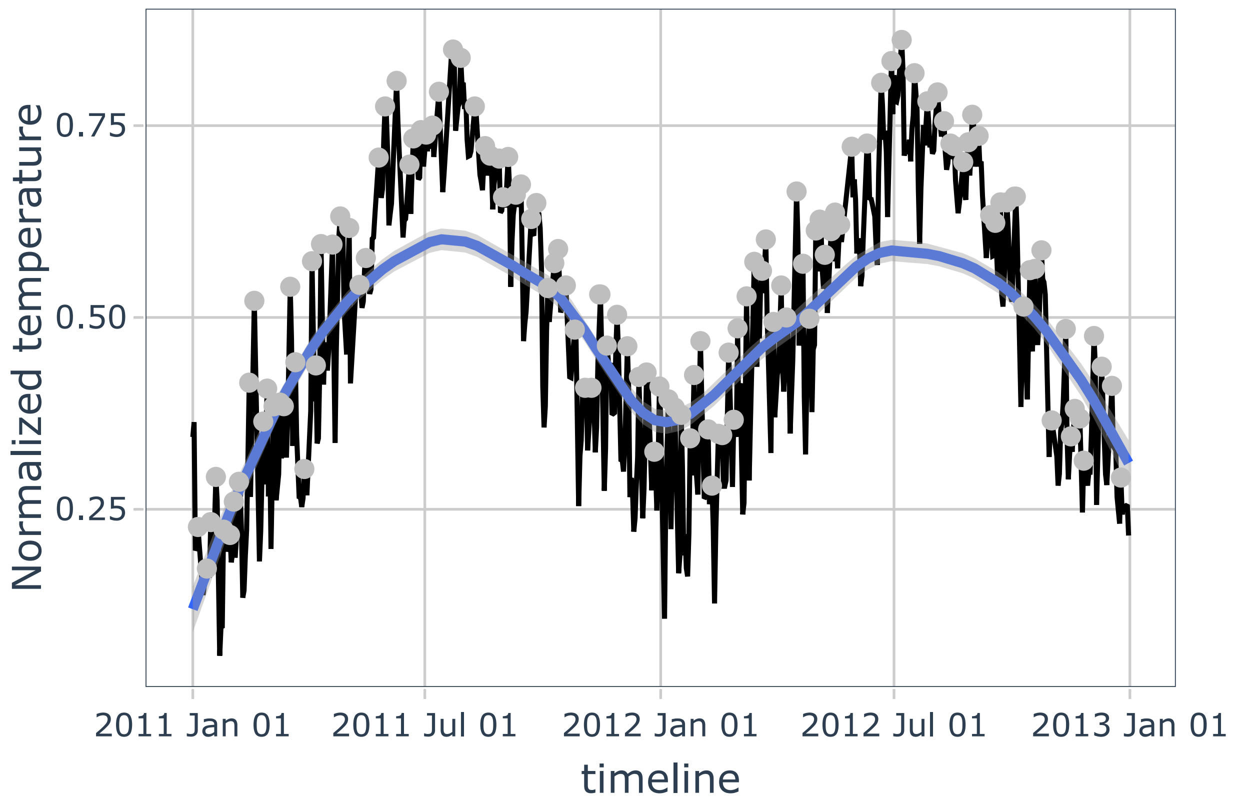
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



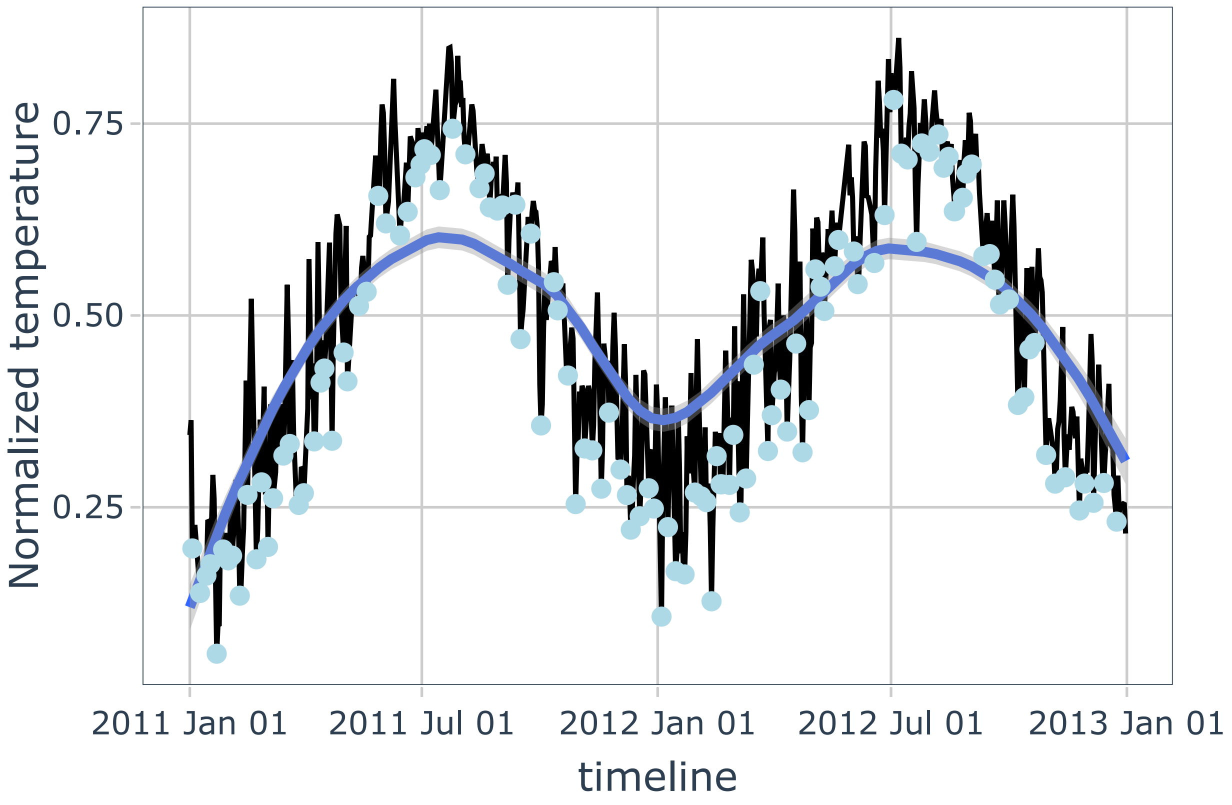
```
# plot peaks
plot_2<-plot_1+stat_peaks(colour = "grey")
ggplotly(plot_2)
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



```r
# plot valleys
plot_3<-plot_1+stat_valleys(colour = "lightblue")
ggplotly(plot_3)
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



For capturing the trend of the data, we fit smoothing conditional means for stat_smooth to figure out if this option visually captures some trend of the data.

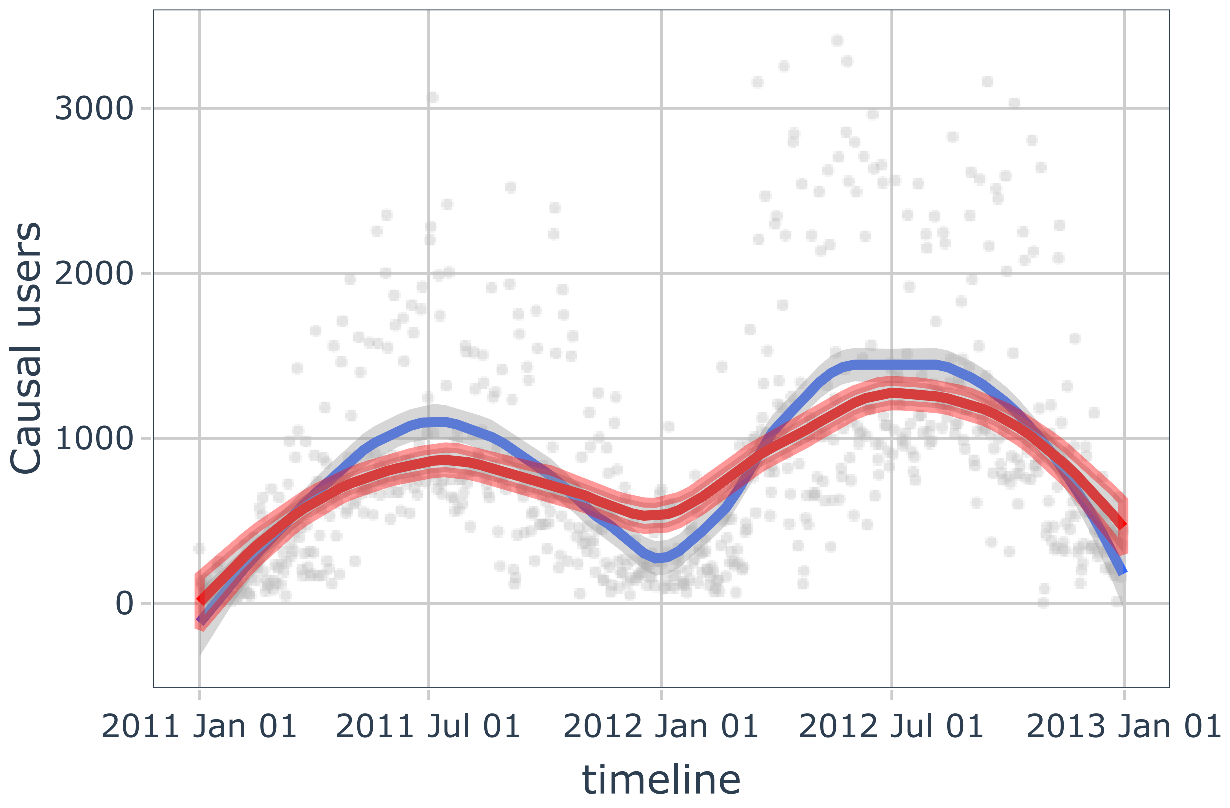
By plotting two different options, local minima and local maxima of the time series data, we want to know if this trend curve of smoothing conditional means is somewhat captured by the points of local minima and maxima.

However, we may find from this visually smoothed curve that there is a trend for seasonality in the normalized temperature where the drastic decrease happens in Winter as to know with.

```
# casual, count of users
df<-data.frame(Casual = dat[,14], Timeline)
colnames(df)<-c('Casual')
plot_4<-ggplot(df, aes(Timeline, Casual)) + geom_point(na.rm=TRUE, color='grey', size=0.7, alpha=0.4)+
# by geom_smooth loess on time series data, compare for difference smoothe estimator
ggplotly(plot_4)
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

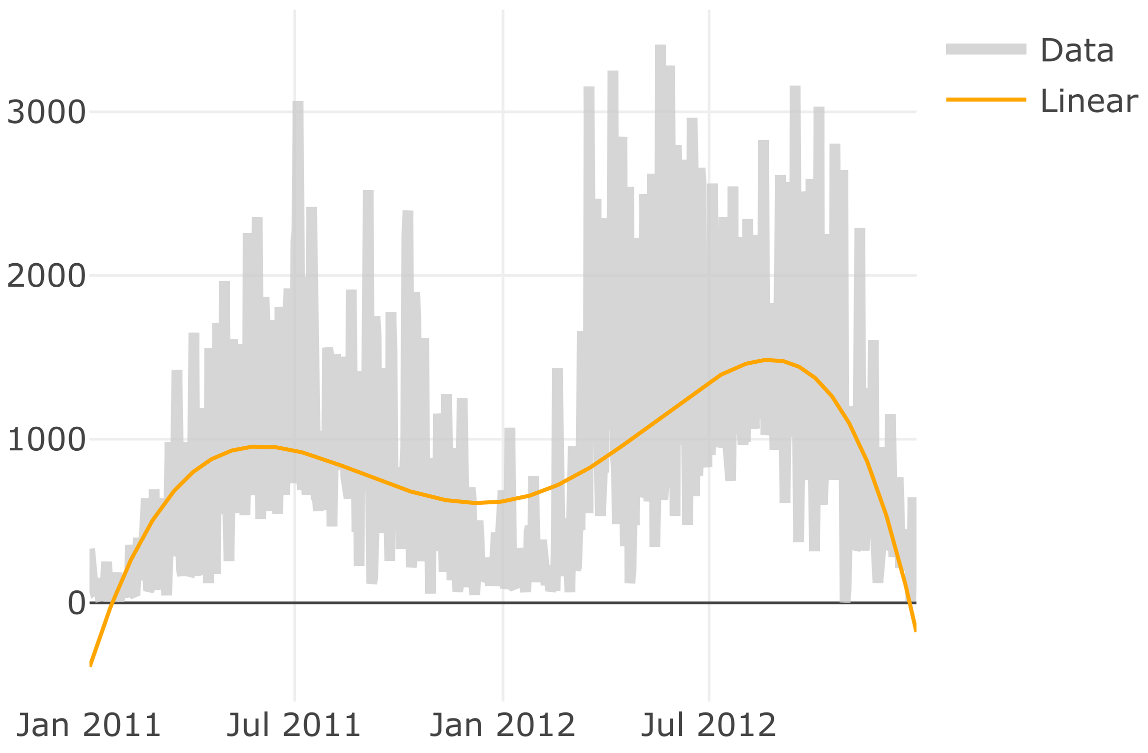


For the consumer data where there is more variability, we may also fit two different curves, to capture is there is any curve that could possibly capture the variability to account for the points.

However, we may find from this visually smoothed curve of smoothed conditional means captures the trend and more variability than the other for the seasonality in the number of causal consumers where the drastic decrease happens in the Winter also to verify with.

Moreover, it passes through more points than the other as well as find some adequate median number of causal users in the data to match with the time trained data.

```
# linear prediction estimation method
ti = 1:length(df$Casual)
m3 = lm(df$Casual~ti+I(ti^2)+I(ti^3)+I(ti^4)+I(ti^4)+I(ti^5))
# estimated values
data.fmt=data.frame(color = c("#CCCCCCCC"), width=4)
line.fmt=data.frame(dash = c("solid"), width=1.5, color=c('orange'))
p.glob = plot_ly(y=df$Casual, x=Timeline, type="scatter", mode="lines", line=data.fmt, name="Data")
p.glob = add_lines(p.glob, x=Timeline, y=predict(m3), line=line.fmt, name="Linear")
p.glob %>% layout(
  xaxis = list())
```



This special curve of linear model captures the trend and variability well for the time series data. One of the simplest methods to identify trends is to fit a ordinary least squares regression model to the data.

The model most people are familiar with is the linear model, but we can add other polynomial terms for extra flexibility. In practice, we avoid polynomials of degrees larger than three because they are less stable but the 5 degrees of polynomial model well capture the trend and variability of the model excluding both tails of the polynomial model which goes below 0.

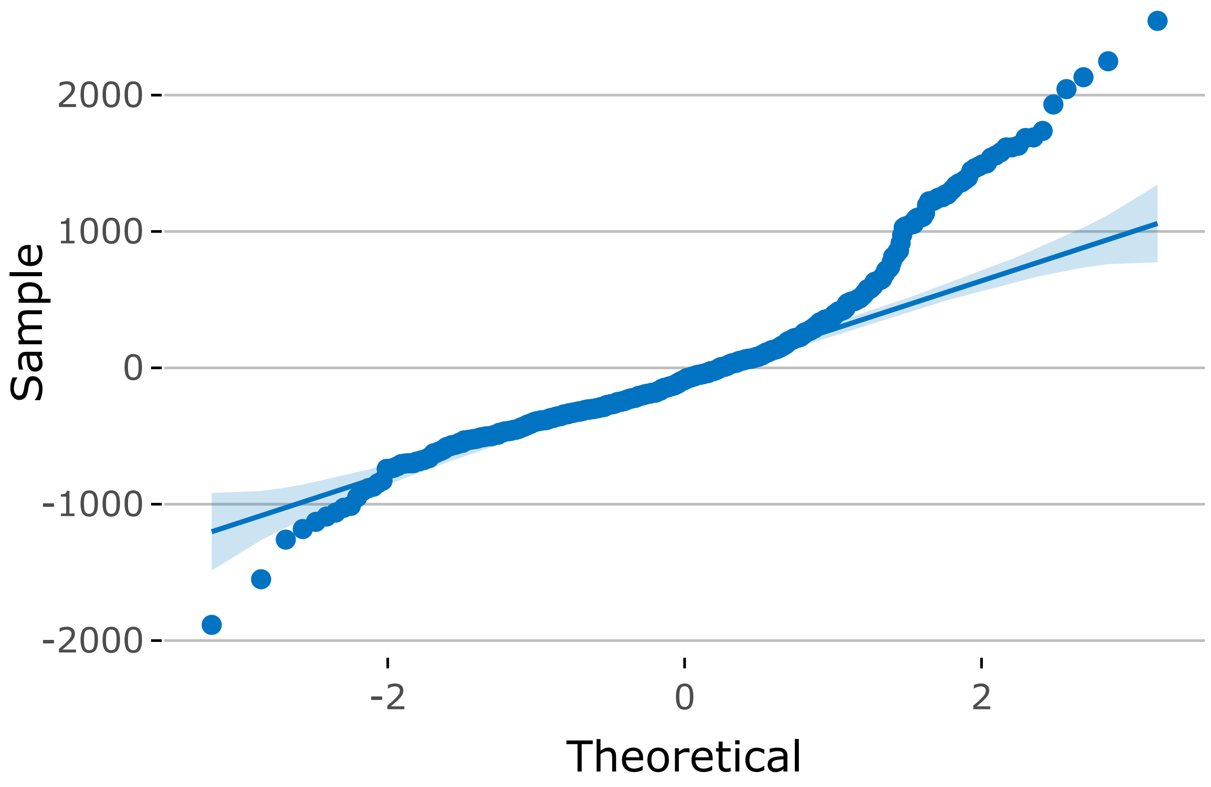
ARMA model for Casual users

```
#prewhiten x
# RGDP residual change rate model
mod.arma <- auto.arima(tm.obs, max.p = 52, stationary = T, seasonal=F, ic = c( "bic"))
kable(coef(mod.arma))
```

	x
ar1	0.7530733
ar2	-0.2693767
ar3	0.1066855
intercept	883.8757102

```
# residuals white noise check
ggplotly(ggqqplot(mod.arma$residuals, color = c("#0073C2FF"),
   ggtheme = theme_pubclean()))
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



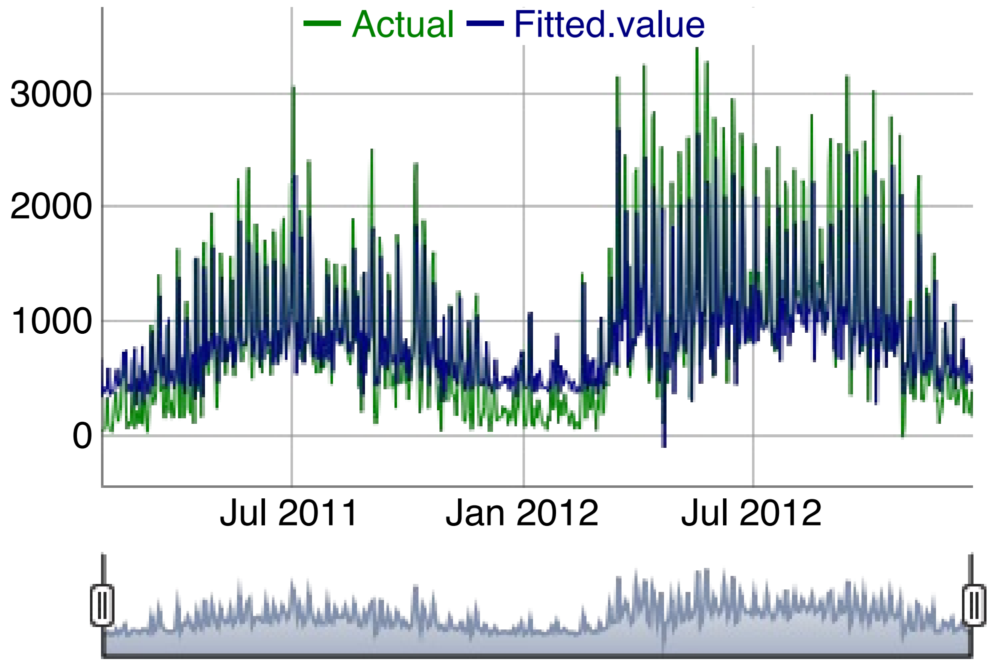
```
# compare arima model and original data
Timeline<-seq(as.Date("2011-02-01"), as.Date("2012-12-24"), by='days')
comb <- cbind(as.vector(tm.obs), as.vector(mod.arma$fitted))
colnames(comb)<-c('Actual', 'Fitted value')
comb<-data.frame(Timeline, comb)
```

We may find the ARMA model for the causal user is not perfectly fitted well by the comparing the actual values and its fitted values.

The model has some deviation for its residuals to deviate as a white noise for its residuals but it also broadly fits for the model of AR(3) from its significant lag by the code of auto.arima which searched for the optimal model by BIC criteria.

However, some minor steps for identification of PACF and ACF plots are skipped as the main goal is the build the TFN model using normalized temperature.

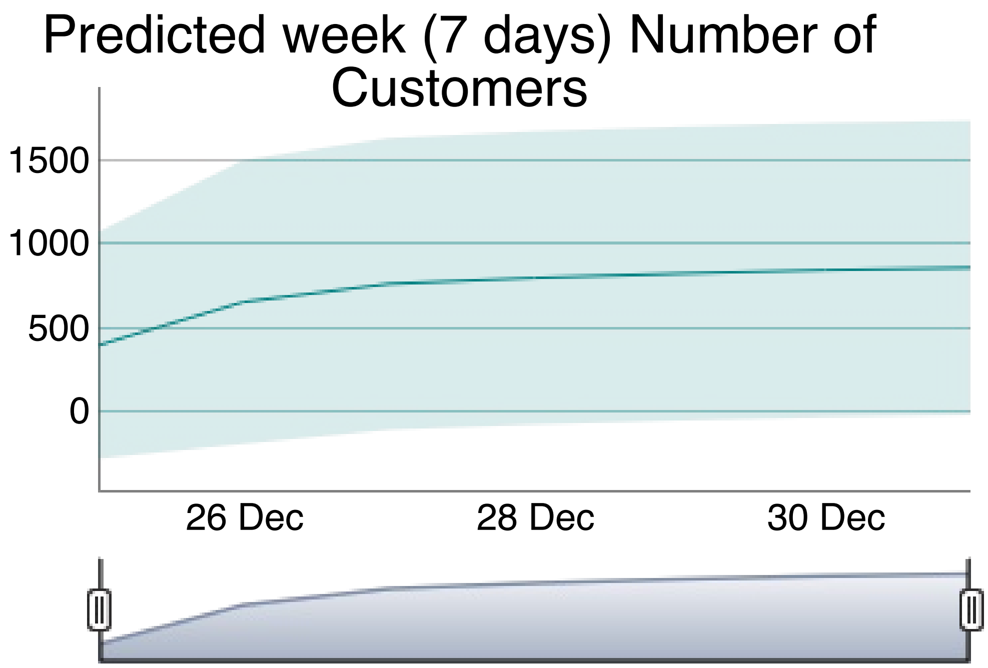
```
comb = xts(x=comb[,-1], order.by=comb$Timeline)
dygraph(comb) %>% dyRangeSelector()
```



```
# forecast
pre<-forecast(mod.arma, h = 7, findfrequency = TRUE)
```

```
## Warning in forecast.forecast_ARIMA(mod.arma, h = 7, findfrequency = TRUE): The
## non-existent findfrequency arguments will be ignored.
```

```
pre_1<-cbind(pre$mean, pre$upper[,1], pre$lower[,1])
colnames(pre_1)<-c('fitted', 'upper', 'lower')
pre_1 = ts(pre_1, start = as.Date("2012-12-25"), end = as.Date("2012-12-31"))
# forecast graph
dygraph(pre_1, main = "Predicted week (7 days) Number of Customers") %>%
  dyAxis("x", drawGrid = FALSE) %>%
  dySeries(c("lower", "fitted", "upper"), label = "Temperature")%>% dyRangeSelector()
```



We can find easily the AR(3) model deviates from the actual points in the Winter season heavily while other seasons are easily fitted against the actual values.

Also, the predicted values with 90% confidence interval could be visualized for practical prediction usage as for reference purpose.

ARMA model of normalized temperature data and fitting plan for TFN model

To Conduct `prewhitening` analysis to identify the lead-lag relationship between number of casual users and noramlized temperature;

- ARMA model for normalized temperature and its residual ACF and PACF plots
- We use cross correlation plot of prewhitened processes to identify transfer function (ν_i)

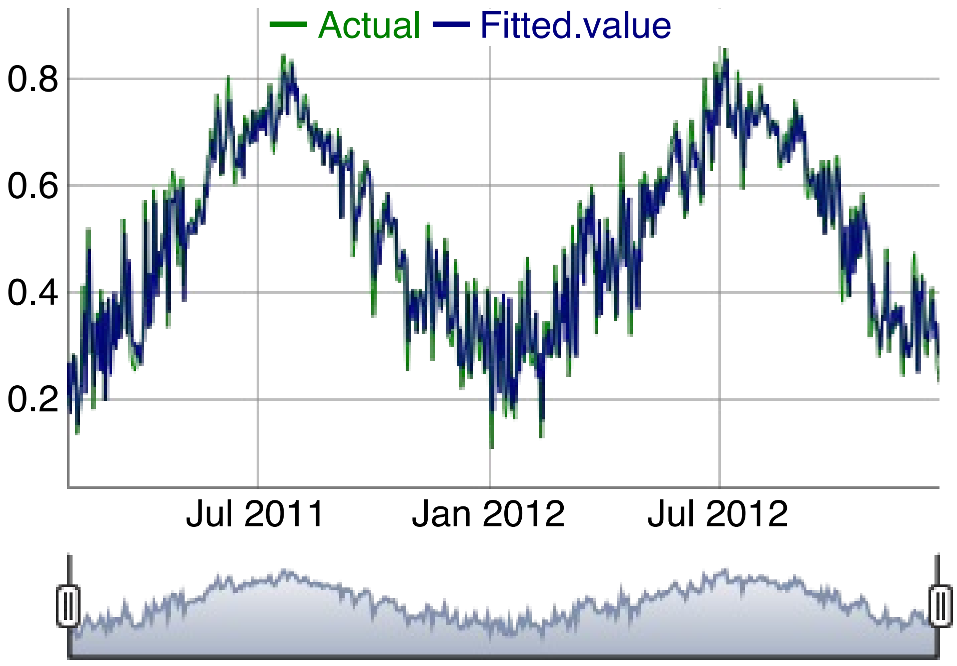
This would lead to give the accommodated model of transfer function noise model for casual users variable eliminating the correlation that exists between two variables of data.

Note that the casual relationship is verified for us to test only for the cross-correlation plot for the daily number of causal users to model against the normalized temperature with significant lag found from the CCR plot.

```
# prewhiten x
# RGDP residual change rate model
mod.arma1 <- auto.arima(temp.obs, max.p = 52, stationary = T, seasonal=F)
kable(coef(mod.arma1))
```

	x
ar1	0.9446582
ar2	-0.3027186
ar3	0.1173646
ar4	0.0371846
ar5	0.0904035
ar6	0.0896740
intercept	0.4589526

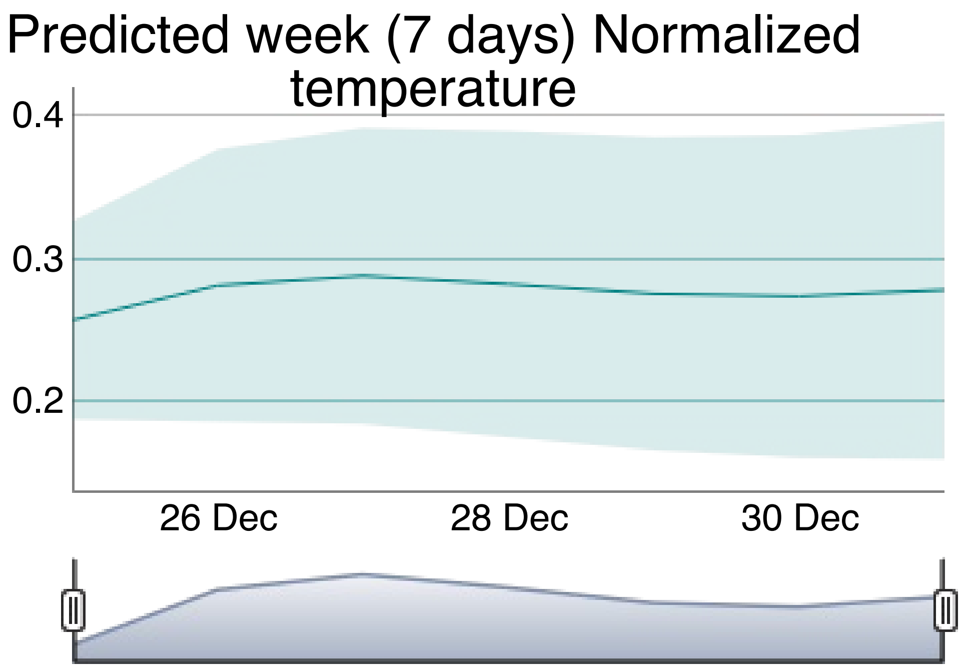
```
## unit root test
par(mfrow=c(1,2))
plot(arroots(mod.arma1), main="Inverse AR roots")
# compare arima model and original data
compa <- cbind(as.vector(temp.obs), as.vector(mod.arma1$fitted))
colnames(compa)<-c('Actual', 'Fitted value')
Timeline<-seq(as.Date("2011-02-01"), as.Date("2012-12-24"), by='days')
compa<-data.frame(Timeline, compa)
compa = xts(x=compa[,-1], order.by=compa$Timeline)
dygraph(compa) %>% dyRangeSelector()
```

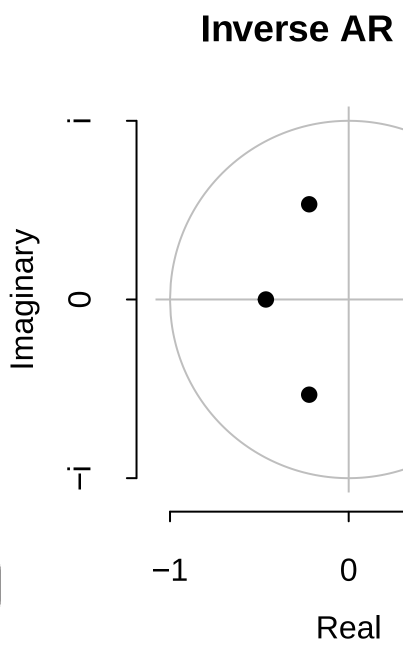


```
# forecast
pre<-forecast(mod.arma1, h = 7, findfrequency = TRUE)
```

```
## Warning in forecast.forecast_ARIMA(mod.arma1, h = 7, findfrequency = TRUE): The
## non-existent findfrequency arguments will be ignored.
```

```
pre_1<-cbind(pre$mean, pre$upper[,1], pre$lower[,1])
colnames(pre_1)<-c('fitted', 'upper', 'lower')
pre_1 = ts(pre_1, start = as.Date("2012-12-25"), end = as.Date("2012-12-31"))
# forecast
dygraph(pre_1, main = "Predicted week (7 days) Normalized temperature") %>%
  dyAxis("x", drawGrid = FALSE) %>%
  dySeries(c("lower", "fitted", "upper"), label = "Temperature")%>% dyRangeSelector()
```



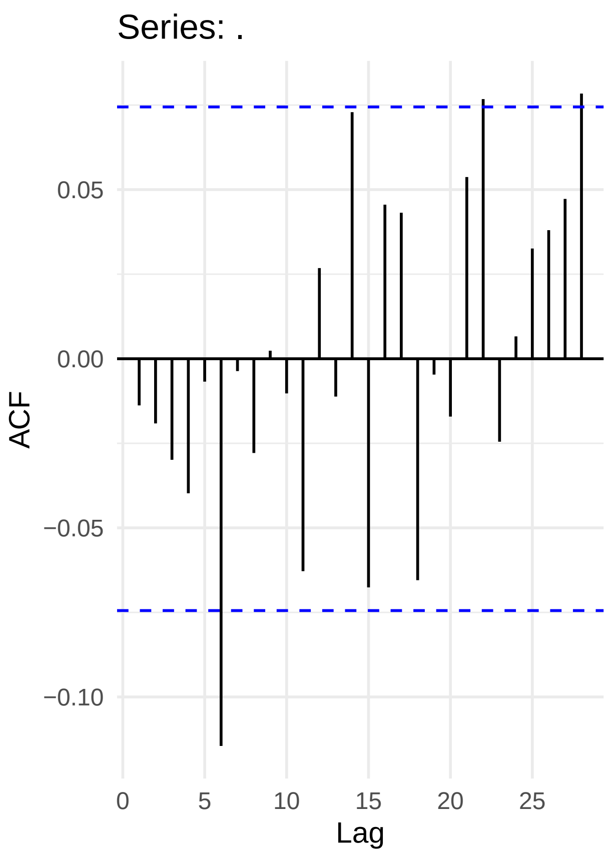


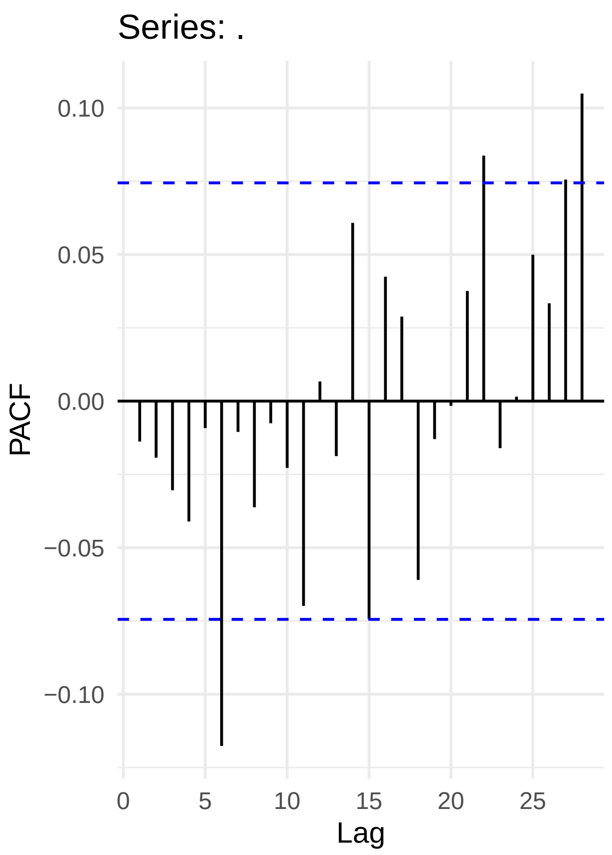
Unlike the time series model of number of causal users, the AR(6) model for the normalized temperature perfectly fits.

This indicates us to easily go on to the next step of verification for the significant lag in TFN model which could be used for the predicting the last 7 days of 2012 in number of causal users.

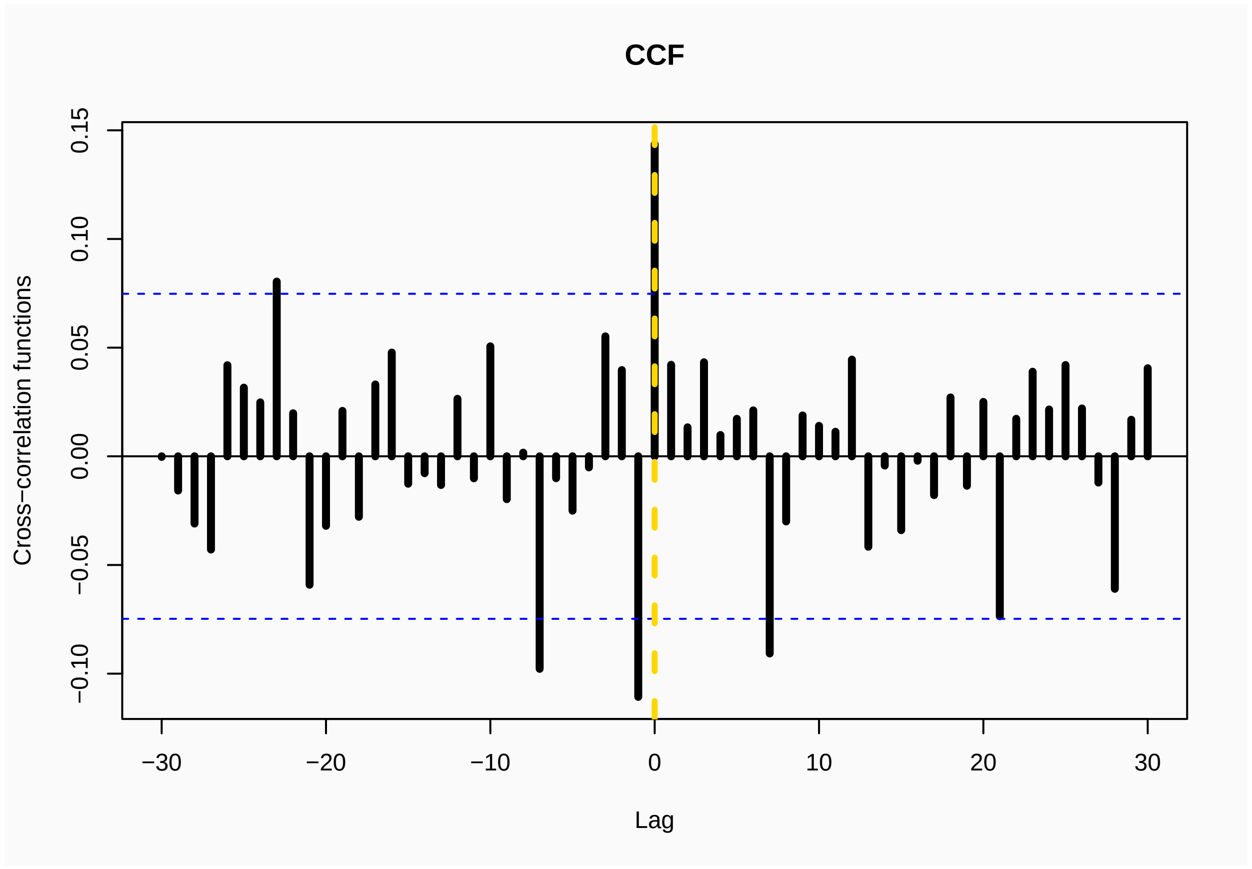
Model this relationship using the transfer function noise model

```
#prewhiten x
# residual check
res<-mod.arma1$residuals
pr = res%>%ggAcf()+theme_minimal()
prs = res%>%ggPacf()+theme_minimal()
grid.arrange(pr,prs, nrow=1)
```





```
# Ljung-Box portmanteau test
p = mod.arma1$arma[1]; q = mod.arma1$arma[2]
temp<-LBTest(mod.arma1$residuals, nPQ = p+q, m = 24, ifPlot = TRUE)
#prewhiten y
mod = mod.arma1;nAR = mod$arma[1]; nMA = mod$arma[2]
if(nMA!=0){
  xf = PreWhiten.arma(temp.obs, ar = mod$coef[1:nAR],
                      ma = mod$coef[(1:nMA)+nAR])[-(1:nAR)]
  yf = PreWhiten.arma(tm.obs, ar = mod$coef[1:nAR],
                      ma=mod$coef[(1:nMA)+nAR])[-(1:nAR)]
}else{
  xf = PreWhiten.arma(temp.obs, ar = mod$coef[1:nAR],
                      ma = 0)[-(1:nAR)]
  yf = PreWhiten.arma(tm.obs, ar = mod$coef[1:nAR],
                      ma=0)[-(1:nAR)]
}
#ccf plot prewhiten x and y
par(cex=0.75, bg='gray98')
ccf(c(xf), c(yf), lwd=4, ylab="Cross-correlation functions",lag.max=30,
    main="CCF")
abline(v=0, col="gold", lwd=3, lty="dashed")
text(-1, 0.2, '-1', col=2);text(-2, 0.2, '-2', col=2)
```



The residuals is observed to be white noise for stationary process with PACF and ACF plot except for the significant lag at 6 as to be AR(6) model without any patterns and also have root inside the unit circle. Also, the residual of the model passes the Ljung-Box portmanteau test where all values are greater than 0.05.

Observing from the Cross-Correlation plot of prewhitened number of causal users and the normalized temperature, the prewhitened number of causal user is constructed by lagged regression (relationship) of the normalized temperature as stated in Granger causality test to be predicted for significant negative lags from -1 to 0 for its *most significant lag* (higher than the other and significantly large) which is used.

Note that the plot of lags reflect the actual significant and we seek for the most significant lag along with its following lags in the negative sides, not positive side (which means the temperature is the result of causality), to choose for the sequence significant lags.

1 Fit a multiple regression using the findings in the `prewhitening` step, i.e.

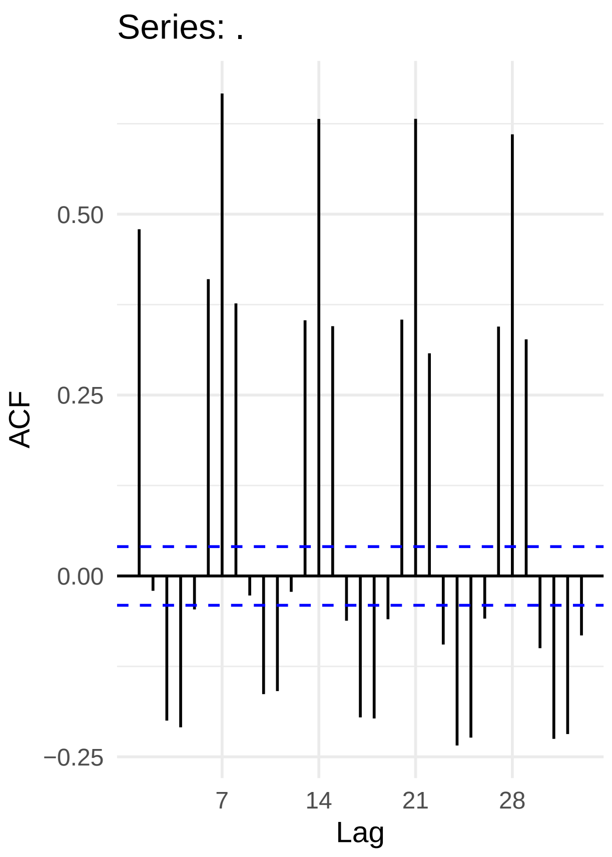
$$y_t = \sum_i v_i x_{t-i} + \xi_t, \quad (1)$$

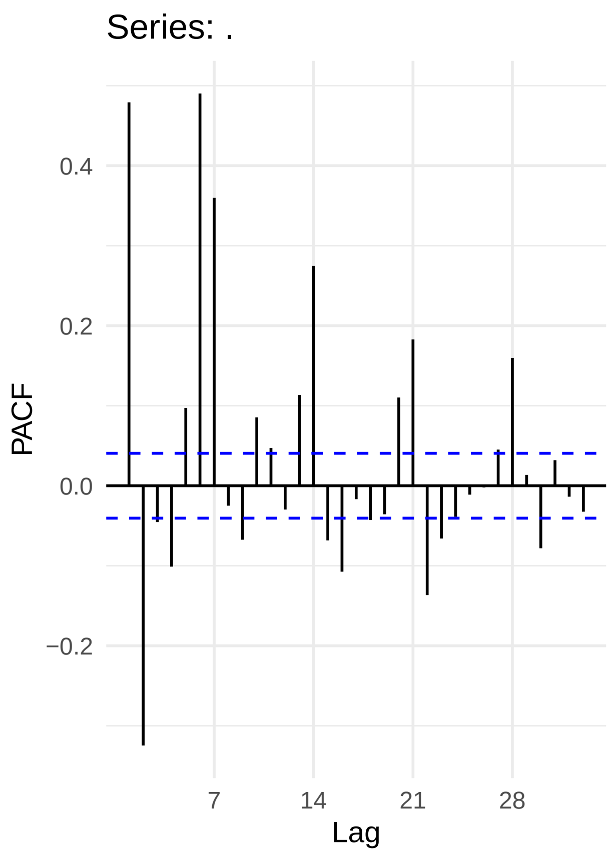
where y_t and x_t denote the output and input process, respectively, and ξ_t is the noise process. We had used `prewhitening` to select the lagged $\{x_i\}$ in the regression

```r
#fit Equation (1)
CU2 = ts(dat[,14], frequency = 7, start = as.Date("2012-02-01"), end = as.Date("2012-12-31"))
TEM = ts(dat[,10], frequency = 7, start = as.Date("2012-02-01"), end = as.Date("2012-12-31"))
mod.dynlm = dynlm(CU2~L(TEM, 0:1))
kable(t(coef(mod.dynlm)))
```

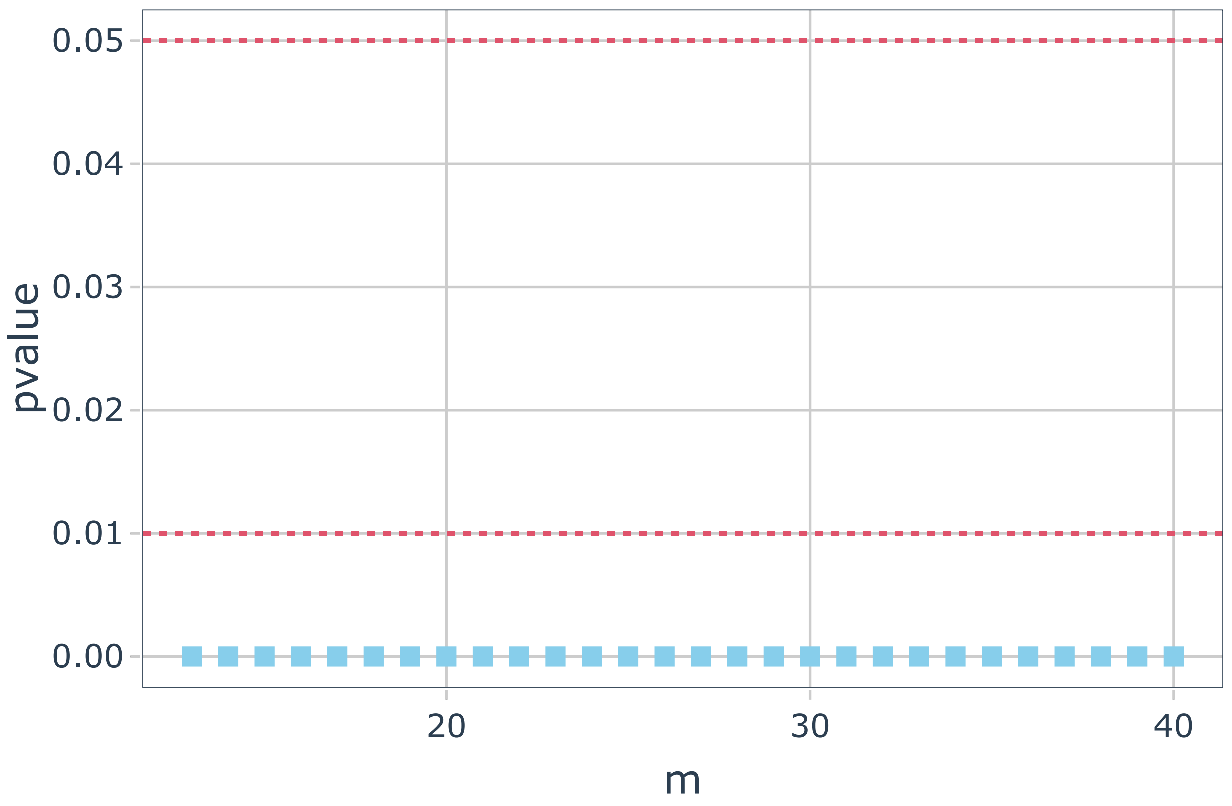
(Intercept)	L(TEM, 0:1)0	L(TEM, 0:1)1
-179.4885	2069.249	-8.044983

```
#plot residual ACF and PACF of the above regression
res1<-ts(mod.dynlm$residuals, frequency = 7, end = c(2012,12,31))
pr1 = res1%>%ggAcf()+theme_minimal()
prs1 = res1%>%ggPacf()+theme_minimal()
grid.arrange(pr1,prs1, nrow=1)
```



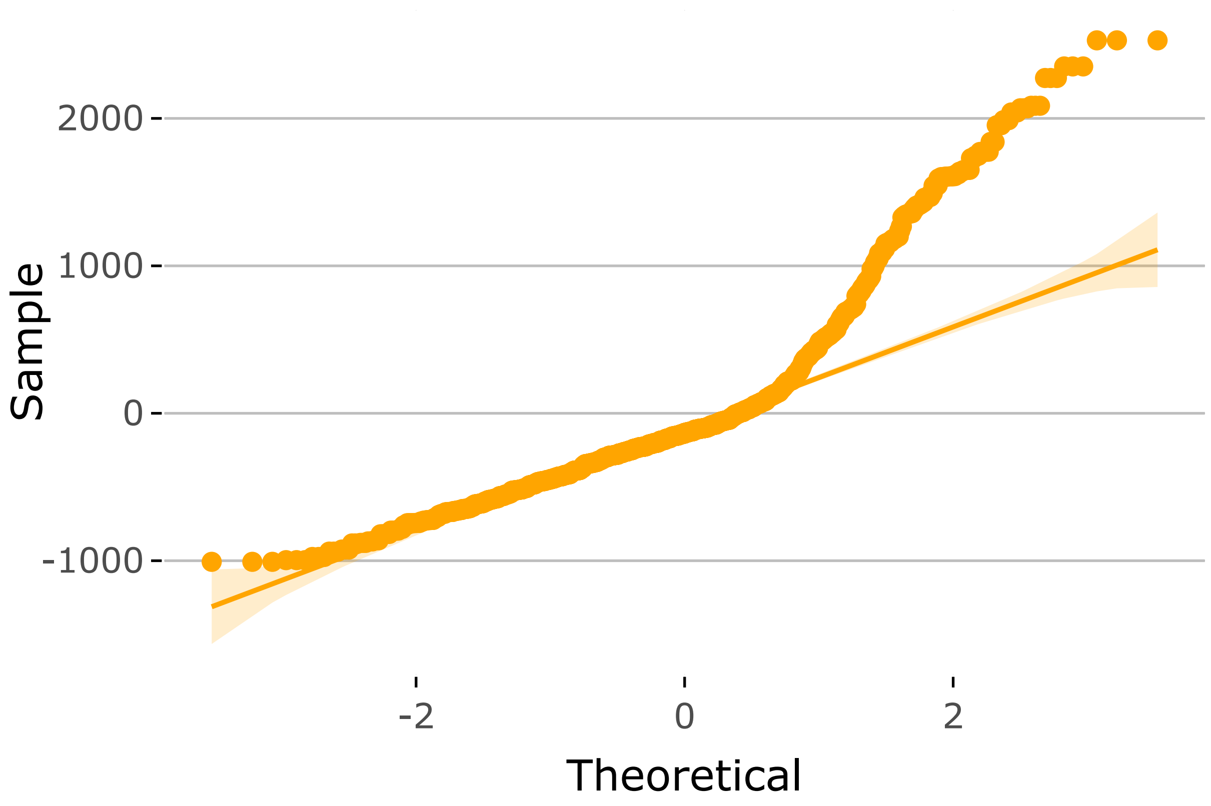


```
# Ljung-Box portmanteau test
LBTest(mod.dynlm$residuals, nPQ = 12, m = 40, ifPlot = TRUE)
```



```r
# Alternative QQplot
ggplotly(ggqqplot(mod.dynlm$residuals, color = c("orange"),
                  ggtheme = theme_pubclean()))
```

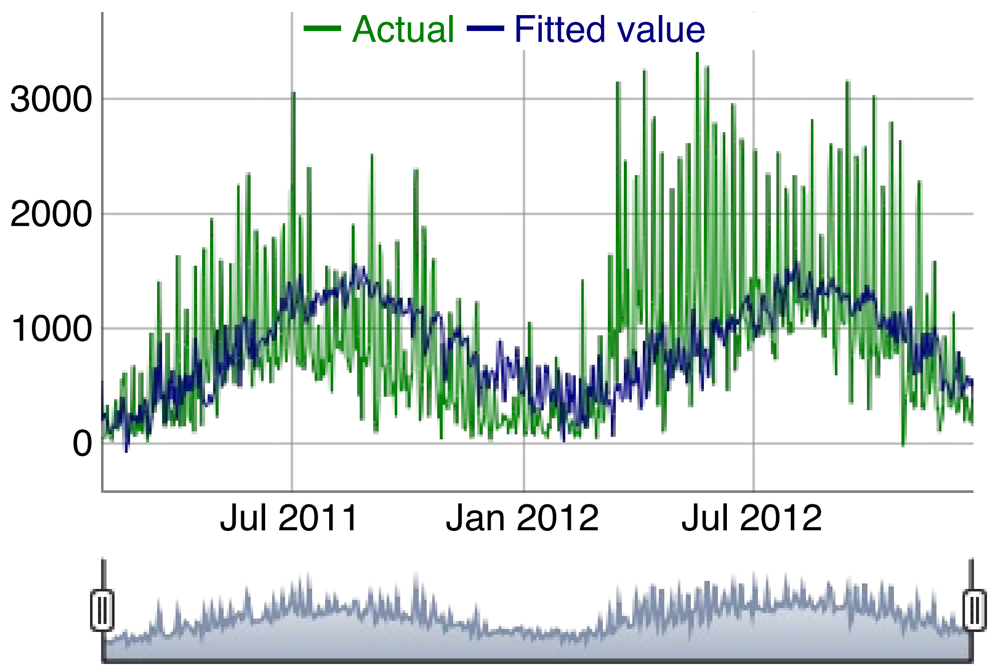
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



```
# Alternative: Shipiro-Wilk test
shapiro.test(mod.dynlm$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  mod.dynlm$residuals
## W = 0.86154, p-value < 2.2e-16
```

```
# failed model:  compare arima model and original data
combi <- cbind(tm.obs, mod.dynlm$fitted.values[1:length(tm.obs)])
colnames(combi)<-c('Actual', 'Fitted value')
dygraph(combi) %>% dyRangeSelector()
```



Based on the ACF plot, let choose the $p = 6$ and the pacf plot gives $q = 6$ such that the sum of $npq = 12$ with max lag to test in LBjung's test for stationarity.

However, from ACF and PACF plot we can find the model is **not** adequate for multiple regression, as dynamic regression model.

But this is part of the step to reason why we construct the model of TFN, not multiple regression.

The multiple regression has used the number of causal users as response y-values with time train of 2011-02-01 up to 2012-12-31 and the normalized temperature as x-variables, with 0 to -1 significant lags observed from the CCF found.

The model is the number of causal users at time t, and TM_t stands for the normalized temperature at time t, $n_{1,t}$ stands for the noise process for D_t.

However, the multiple regression model is **non-stationary** with serially correlated residuals from PACF and ACF plot with Ljung Box test to be **not** passed (we need to go over the line of threshold).

Ljung Box portmandeu test is not accurate all times but we can also test if it is white noise by the QQplot where there are still much deviation to notice with.

Moreover, the Shapiro-Wilk test for normality shows to reject null hypothesis and conclude that the model is not stationary with small p-value.

Hence, the model could **not** be used.

Fit a transfer function noise model using the rational distributed lag function {-} i.e.

$$y_t = \frac{\delta(B)}{\omega(B)} x_t + n_t$$

where $\delta(B)$ and $\omega(B)$ are polynomials in the *backward shift operator* B, and n_t follows an ARMA process.

We would go through steps to identify the mathematical representation of the fitted model.

```
#fit Equation and show the fitted model
temp.obs = window(temp.dat, start = "2011-02-01", end = "2012-12-31", by='days')
tm.obs = window(del.dat, start = "2011-02-01", end = "2012-12-31", by='days')
x <- temp.obs
```

```r
y <- tm.obs
datt<- cbind(y, x, stats::lag(x, 1))[-c(1),]
colnames(datt)<-c("Cas","Tem","Tem1")
tim0 <-timeSequence(from = "2011-02-02", to = "2012-12-31", by = "day")
data<- timeSeries(datt, charvec = tim0)
## divide into forecast and training dataset
data.obs = window(data, start = "2011-02-02", end = "2012-12-24")
data.test = window(data, start = "2012-12-25", end = "2012-12-31")
# fit TFN model
mod.tfn = auto.arima(data.obs[,1], xreg = data.obs[,-1], seasonal=F, stationary=T)
# coeff
kable(t(round(coef(mod.tfn),2)))
```

ar1	ar2	ma1	intercept	Tem	Tem1
0.79	-0.4	-0.17	-137.91	2253.1	-247.81

For ARMA(2,1) model, it is expressed as,

$$y_t = -137.909 + 0.793y_{t-1} - 0.401y_{t-2} + 2253.103x_t - 247.808x_{t-1} + a_t - 0.173a_{t-1}$$

, where

$$a_t \sim NID(0, 240083)$$

for white noise.

Using compact notation (AR, MA part),

$$(1 + 0.793B - 0.401B^2)y_t = -137.909 + (+2253.103 - 247.808B)x_t + (1 - 0.173B)a_t$$

.

Equivalently,

$$y_t = -99.07 + \frac{(2253.103 - 247.808B)}{(1 + 0.793B - 0.401B^2)}x_t + \frac{(1 - 0.173B)}{(1 + 0.793B - 0.401B^2)}a_t$$

, where

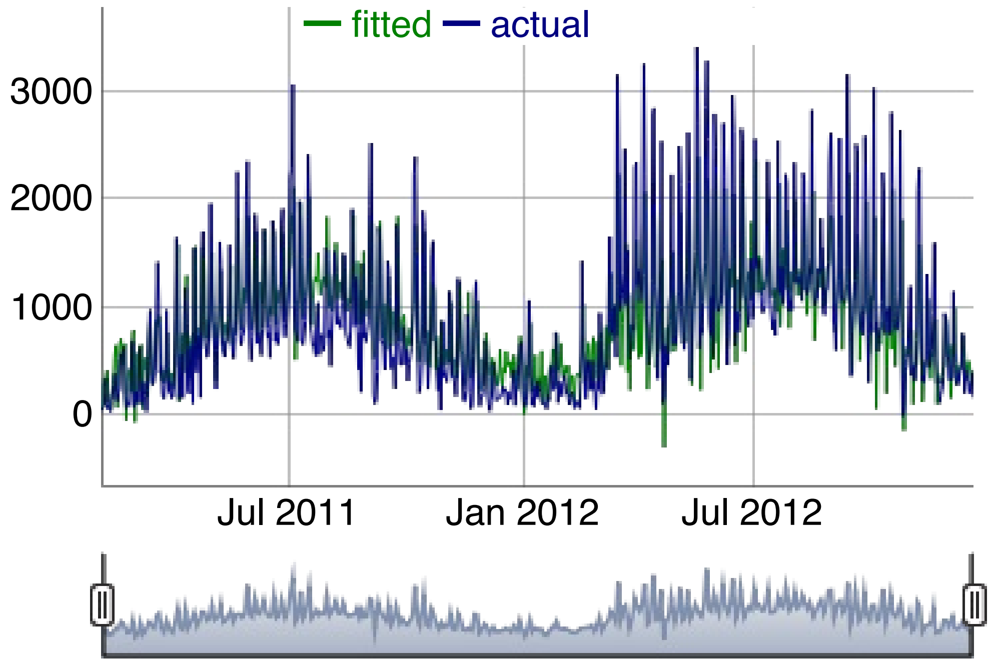
$$a_t \sim N(0, 240083)$$

for white noise and

$$\frac{-137.909}{(1 + 0.793 - 0.401)} \approx -99.07$$

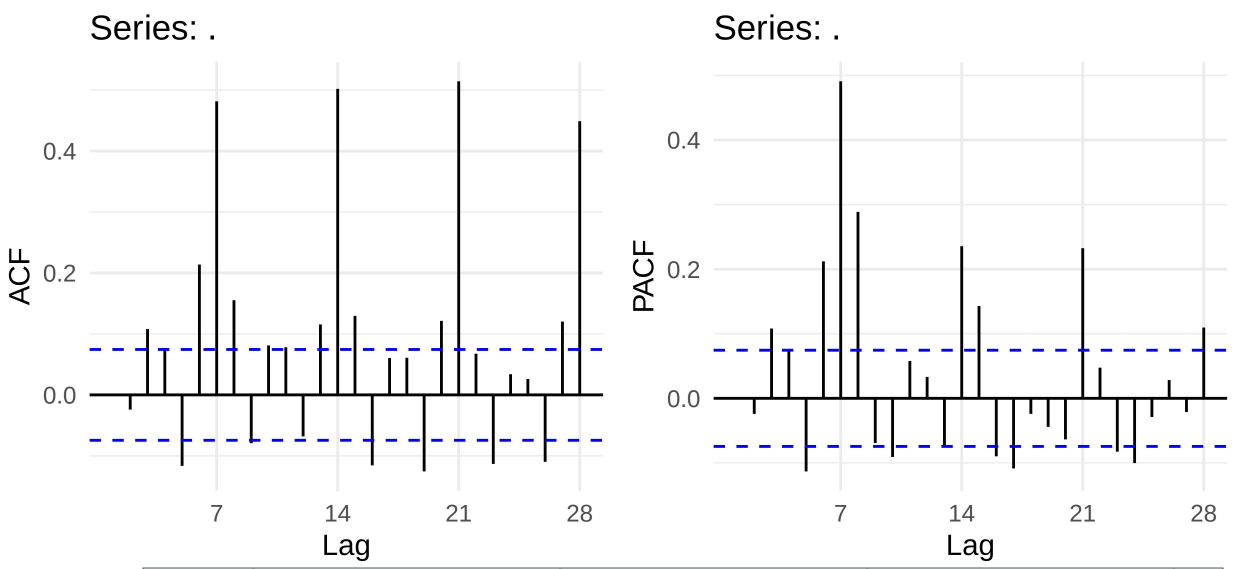
.

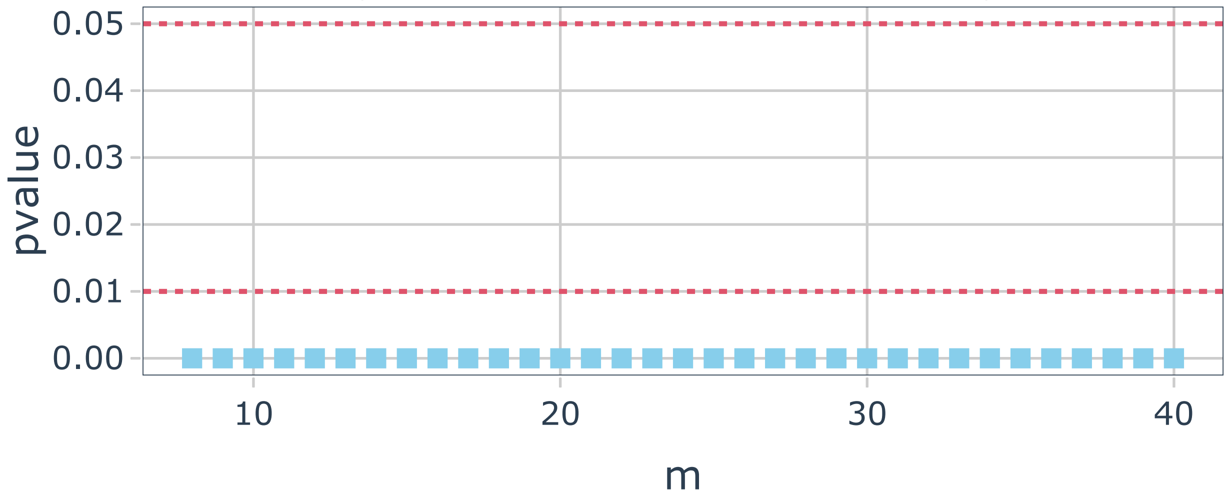
```r
## plot
con<- window(del.dat, start = "2011-02-02", end = "2012-12-24", by='days')
combin<-cbind(mod.tfn$fitted, con)
colnames(combin)<-c('fitted','actual')
# plot
Timeline<-seq(as.Date("2011-02-02"), as.Date("2012-12-24"), by='days')
don <- data.frame(Timeline,combin)
don=xts( x=don[,-1], order.by=don$Timeline)
dygraph(don) %>% dyRangeSelector()
```



However, the model fit it **not** perfect and some part April 23rd 2012 to be below 0 for inadequate point as well. Compare to multiple regression the overall fits and model diagnostic is good to use as a model. For clear comparison, we would use the ARIMA model to compare for the MSE, MAPE, MAE for 7 days left for week for December 25th to 31st.

The model adequacy tests (diagnostics) on the above models





```
##
##  Shapiro-Wilk normality test
##
## data:  mod.tfn$residuals
## W = 0.89478, p-value < 2.2e-16
```

The Ljung-Box test is higher than 0.05 p-value for residuals to **not** pass the test. The blue line of limits are based on the approximate large sample standard error that applies to a white noise process. The sample ACF values exceed these rough critical values is not counted, as the true autocorrelations are both zero.

The plot demostrates asymptotically increasing over m for moderately large enough values for greater p-values which should indicate to be significant after $m = 32$ as to pass the Cross-corrleation plot (by central limit theorem applied).

However, the model diagnostic of TFN model is not passed to fail for further inference. At the end, neural network model could be fitted for most accurate prediction.

```
#forecast using tfn
pre.tfn<-forecast(mod.tfn, xreg = data.test[,-1])
kable(pre.tfn)
```

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
693	363.9009	-264.0373	991.8391	-596.4477	1324.250
694	320.5454	-418.4397	1059.5305	-809.6348	1450.726
695	399.4648	-341.7283	1140.6580	-734.0923	1533.022
696	396.7840	-352.6564	1146.2245	-749.3863	1542.954
697	381.3951	-376.1976	1138.9878	-777.2429	1540.033
698	373.9064	-384.9304	1132.7432	-786.6343	1534.447
699	279.0111	-479.8915	1037.9138	-881.6303	1439.653

```
#calculate MSE, MAE, MAPE
yobs<-as.data.frame(data.test)[,1]
yhat<-forecast(mod.tfn, xreg=data.test[,-1], data.test[,1])$mean
RMSE_TFN<-mean((yobs-yhat)^2)%>% sqrt()
MAE_TFN<-mean(abs(yobs-yhat))
MAPE_TFN<-mean(abs(1-yhat/yobs))
kable(cbind(RMSE_TFN, MAE_TFN, MAPE_TFN))
```

RMSE_TFN	MAE_TFN	MAPE_TFN
193.6001	168.5165	5.368659

2 Sample forecasts of the above fitted models using the remaining observations.

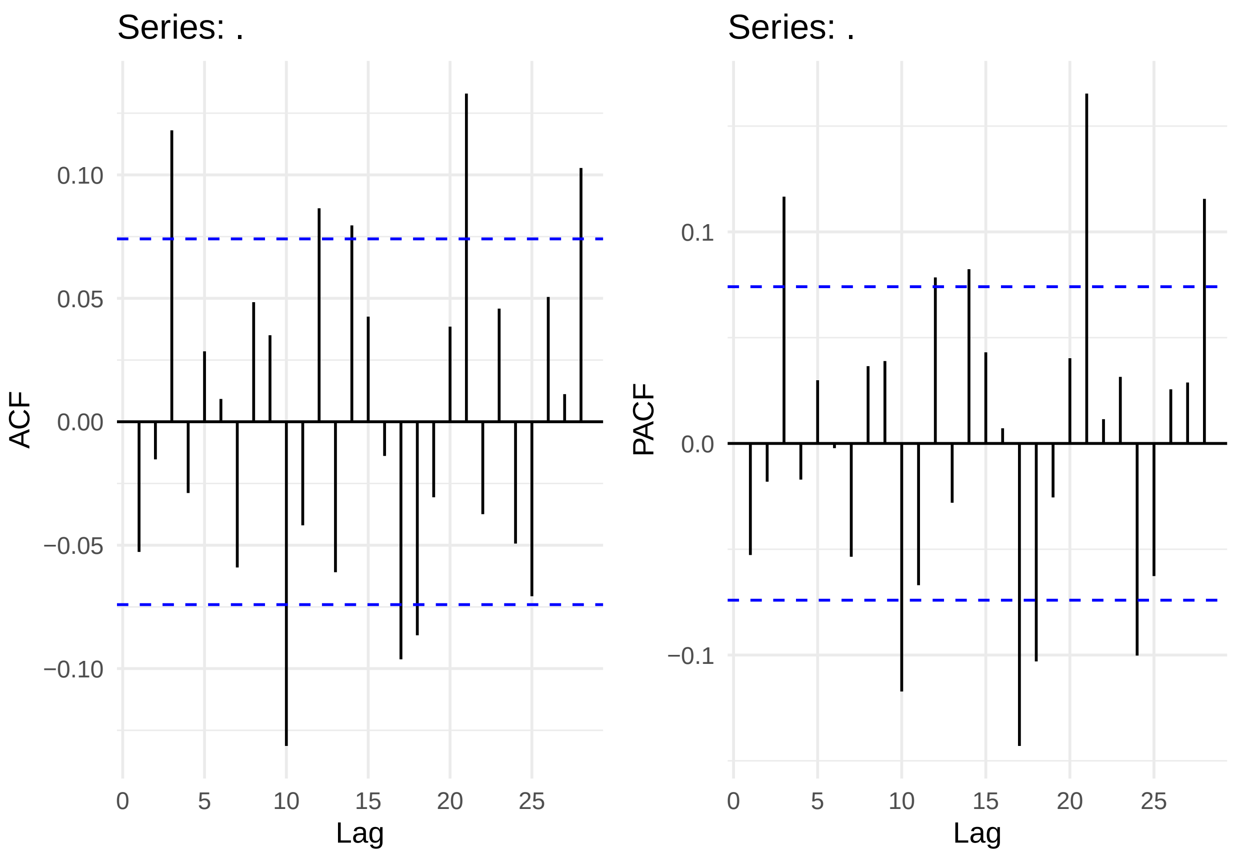
The forecast performance using Mean squared error (MSE), Mean absolute error (MAE), and Mean absolute percentage error (MAPE) is as follows: $RMSE = \sqrt{\frac{\sum_{i=1}^{L}(y_{t+i}-\hat{y}_t(i))^2}{L}}$,
$MAE = \frac{\sum_{i=1}^{L}|y_{t+i}-\hat{y}_t(i)|}{L}$,
$MAPE = \frac{1}{L}\sum_{i=1}^{L}\left|1-\frac{\hat{y}_t(i)}{y_{t+i}}\right|$,
where $\hat{y}_t(i)$ denotes the forecast at origin t with lead time i.

(The MAE is lower the better for the forecast is. The models which try to minimize MAE lead to forecast median. The models trying to minimize RMSE lead to a forecast of the mean. Both MAE and RMSE are scale-dependent errors. This means that both errors and data are on the same scale.)

The same out of sample forecasts soley on y_t using an ARIMA model

We are to compare and discuss its peformance metrics with the TFN model. We may fit an ARIMA model on y_t using `auto.arima` but ensure that the fitted model pass the Ljung-Box test.

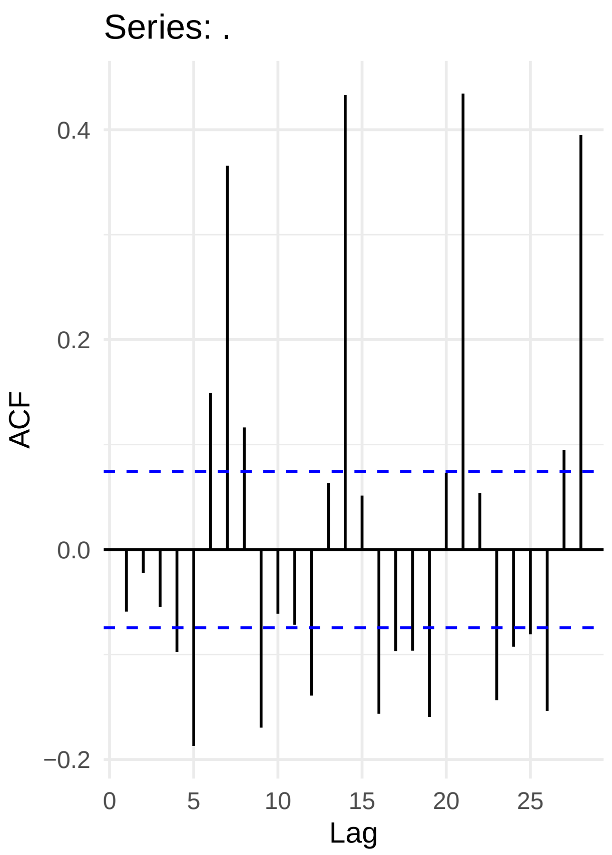
```
#forecast using auto.arima
# PACF, ACF of auto.arima
m<-auto.arima(tm.obs)
pr3 = m$residuals%>%ggAcf()+theme_minimal()
prs3 = m$residuals%>%ggPacf()+theme_minimal()
grid.arrange(pr3,prs3, nrow=1)
```

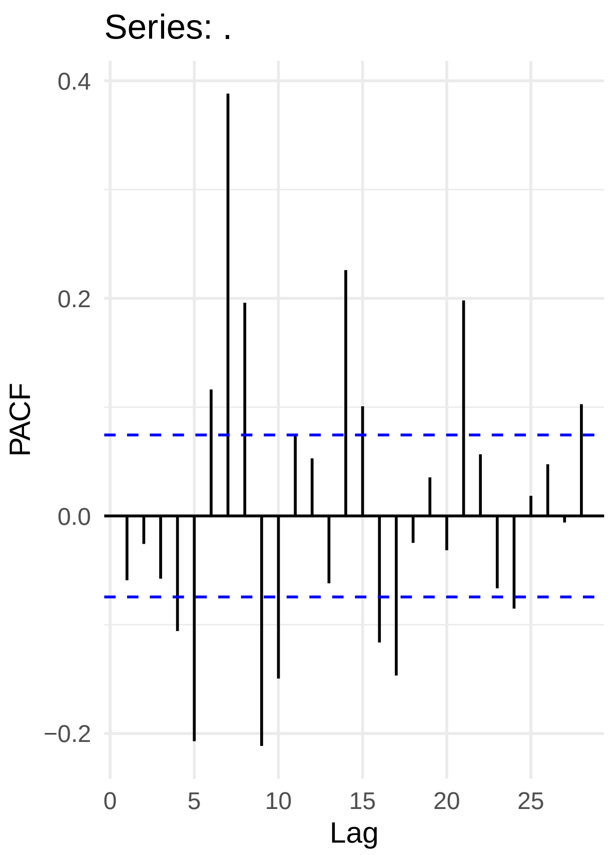


```
# ARIMA
tm.obs = window(del.dat, start = "2011-02-01", end = "2012-12-24")
tm.test = window(del.dat, start = "2012-12-25", end = "2012-12-31")
mod.arima = arima(tm.obs, c(5,0,3))
kable(t(round(coef(mod.arima),2)))
```

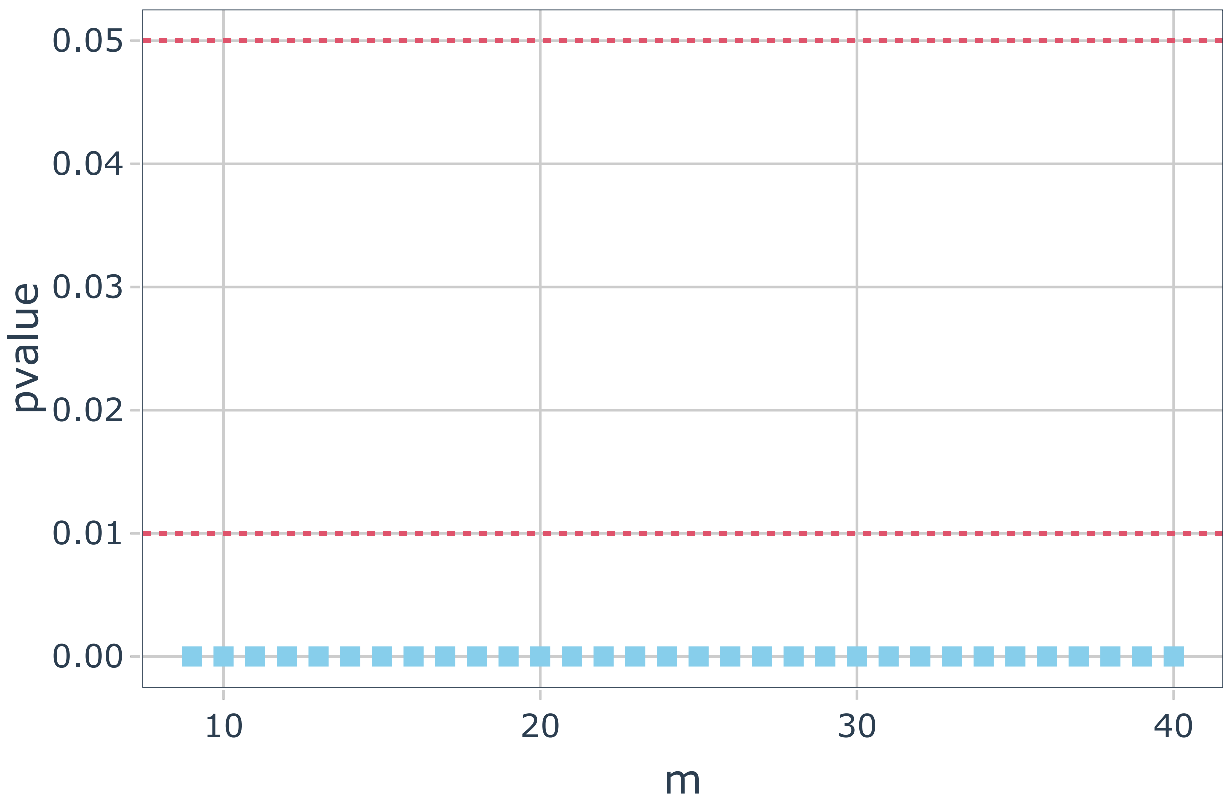
ar1	ar2	ar3	ar4	ar5	ma1	ma2	ma3	intercept
0.33	-0.02	0.78	-0.45	0.35	0.32	-0.13	-0.92	725.97

```
# PACF, ACF, no significant lags
m<-mod.arima$residuals
pr3 = m%>%ggAcf()+theme_minimal()
prs3 = m%>%ggPacf()+theme_minimal()
grid.arrange(pr3,prs3, nrow=1)
```

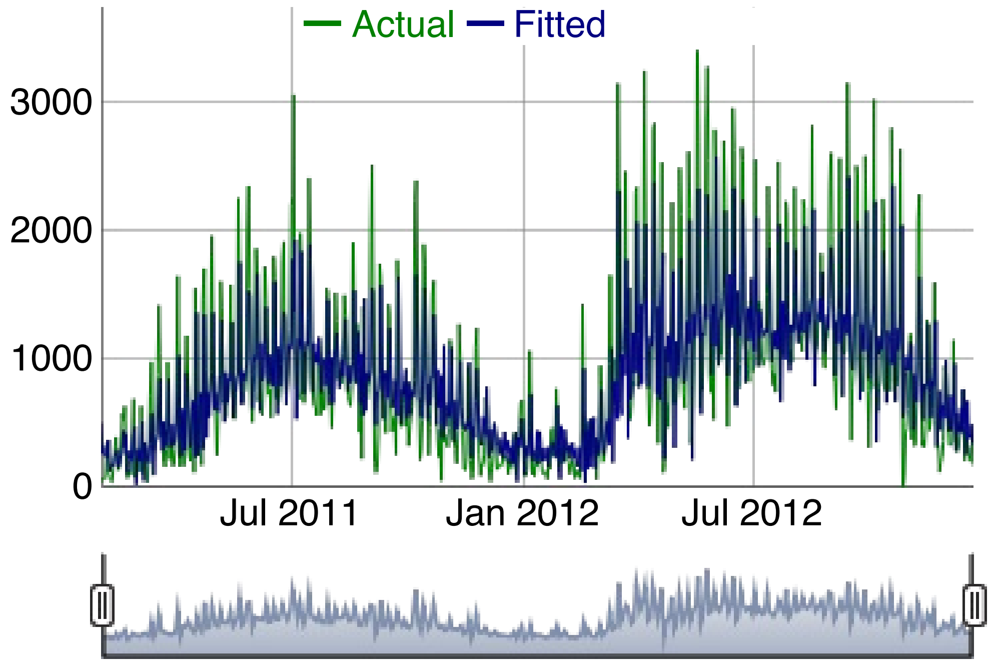




```
par(mfrow = c(1,1))
## Ljung test
p = mod.arima$arma[1]; q = mod.arima$arma[2]
LBTest(mod.arima$residuals, nPQ = p+q, m = 40, ifPlot = TRUE)
```



```
# combine for data formation
combina <- cbind(as.vector(tm.obs), fitted(mod.arima))
colnames(combina)<-c('Actual','Fitted')
# plot
Timeline<-seq(as.Date("2011-02-01"), as.Date("2012-12-24"), by='days')
contr_pl <- data.frame(Timeline, combina)
contr_pl = xts( x=contr_pl[,-1], order.by=contr_pl$Timeline)
dygraph(contr_pl) %>% dyRangeSelector()
```



```
# forecast 7 points
pre.arima<-forecast(mod.arima, h=7)
kable(pre.arima)
```

	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
694	382.2978	-231.8108	996.4064	-556.9003	1321.496
695	426.3915	-304.0999	1156.8829	-690.7987	1543.582
696	465.2687	-266.0273	1196.5647	-653.1520	1583.689
697	516.4184	-219.2528	1252.0896	-608.6936	1641.530
698	392.6156	-343.0845	1128.3157	-732.5406	1517.772
699	433.1903	-305.4228	1171.8033	-696.4209	1562.801
700	487.2009	-255.7352	1230.1371	-649.0219	1623.424

```
#calculate MSE, MAE, MAPE
yobs<-as.data.frame(data.test)[,1]
yhat<-forecast(mod.arima, h=7)$mean
RMSE_ARMA<-mean((yobs-yhat)^2)%>% sqrt()
MAE_ARMA<-mean(abs(yobs-yhat))
MAPE_ARMA<-mean(abs(1-yhat/yobs))
kable(cbind(RMSE_ARMA, MAE_ARMA, MAPE_ARMA))
```

RMSE_ARMA	MAE_ARMA	MAPE_ARMA
208.1038	167.4215	7.051275

```
# compare
ARMA<-rbind(RMSE_ARMA, MAE_ARMA, MAPE_ARMA)
TFN<-rbind(RMSE_TFN, MAE_TFN, MAPE_TFN)
comp<-cbind(ARMA, TFN); rownames(comp)<-c('RMSE', 'MAE', 'MAPE')
colnames(comp)<-c('ARMA','TFN')
kable(comp)
```

	ARMA	TFN
RMSE	208.103802	193.600104
MAE	167.421538	168.516517
MAPE	7.051275	5.368659

The auto.arima model found with ARMA(5,3) is **not** adequate as the residual of PACF and ACF plot shows serially correlated lags above significant level. The model is found with ACF and PACF plot for significant lags and also the manual model had white noise to be not much suited for the residuals.

First, the ARMA model for deliquency rate is ARMA(5,3) where residuals do *not* show any pattern and pass Ljung-Box portmanteau test for over p-value of 0.05. The roots are all contained in the unit circle to be stationary process.

The RMSE and MAPE is higher for ARMA model than the TFN model indicating the fitted model is better in prediction for 7 points from `2012-12-25` to `2012-12-31`.

The MAE value is lower compared to TFN model by approximately 1, indicating the better for the forecast is.

(Simple Ensemble model construction) Although both models of ARMA and TFN is failed, we may construct the combinated model of these two for equal weight 0.5, to forecast and find if there improvement for the forecast.

We are to conduct the same out of sample forecast analysis using forecast combination of the fitted TFN model and ARIMA model (equal weight and MSE weighting).

- *Forecast combination:*
  The combined forecaster $\hat{f}_t(i)$ may be given by

$$\hat{f}_t(i) = w_a\ \hat{y}_t^{(a)}(i) + w_b\ \hat{y}_t^{(b)}(i),$$

  where the superscripts (a) and (b) stand for transfer function noise model and ARIMA model, respectively. For the equal weight scheme, $w_a = w_b = 0.5$, and for the MSE weighting scheme, we may find for the MSE comparison if there is any enhancement in the model.

```
#calculate MSE, MAE, MAPE for the equal weight forecast
yobs<-data.test[,1]
yhat<-forecast(mod.tfn, xreg=data.test[,-1], data.test[,1])$mean
yhat1<-forecast(mod.arima, h=7)$mean
y_w<-as.vector(0.5*yhat)+as.vector(0.5*yhat1)
# MSE
RMSE_w<-mean((yobs-y_w)^2)%>%sqrt()
# MAE
MAE_w<-mean(abs(yobs-y_w))
# MAPE
MAPE_w<-mean(abs(1-y_w/yobs))
kable(cbind(RMSE_w, MAE_w, MAPE_w))
```

RMSE_w	MAE_w	MAPE_w
194.0054	161.0832	6.194282

```
# compare
ARMA<-rbind(RMSE_ARMA, MAE_ARMA, MAPE_ARMA)
TFN<-rbind(RMSE_TFN, MAE_TFN, MAPE_TFN)
same<-rbind(RMSE_w, MAE_w, MAPE_w)
comp0<-cbind(ARMA, TFN, same); colnames(comp0)<-c('ARMA','TFN', 'Same weight')
rownames(comp0)<-c('RMSE', 'MAE', 'MAPE')
kable(comp0)
```

	ARMA	TFN	Same weight
RMSE	208.103802	193.600104	194.005418
MAE	167.421538	168.516517	161.083181
MAPE	7.051275	5.368659	6.194282

The 0.5 equal weight the models give RMSE and MAPE lower than ARMA model prediction but higher than TFN model prediction. However, the MAE is lowered than both models indicating the significance of constructing a ensemble model for time series.

The equal weight of 0.5 for both ARMA and TFN model RMSE, MAE and MAPE is relatively lower by around 14, 6, and 1 but higher than the TFN model RMSE, MAPE and MAE values by around 1, -7 and 1 equivalently.

Clearly, the combined model weighted is **not** low enough to make prediction to be better than the TFN model for 7 forecasting points.

Conclusion

We are necessary to construct the **stationary** model that could fit on robustly for our data. The model diagnostic indicated some failures in construction of the model and it should be resolved for both models. Also, with more forecast points and optimal combination for the ensemble model of TFN and ARMA model, we look forward to fit for better performance model. This would require some more construction on the weight given for both models. However, lastly we can compare for the model performance of the neural network model to find if this would give for the optimal performance.

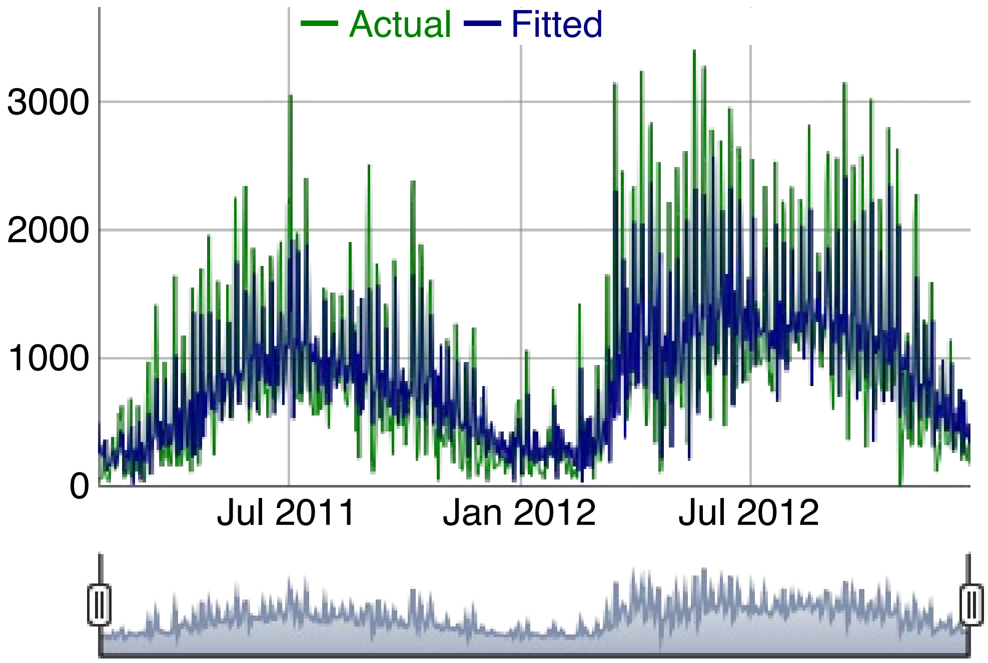
Moreover, considering the seasonality of the year and casual users, we may also use SARIMA models to construct the forecast of different seasonal points.

3 Expansion for simple deep learning models

```r
# Neural network model
mod.nn = forecast::nnetar(tm.obs);mod.nn
```

```
## Series: tm.obs
## Model:  NNAR(28,14)
## Call:   forecast::nnetar(y = tm.obs)
##
## Average of 20 networks, each of which is
## a 28-14-1 network with 421 weights
## options were - linear output units
##
## sigma^2 estimated as 8481
```

```r
# check that roots are within the unit circle
combinat <- cbind(as.vector(tm.obs), as.vector(mod.nn$fitted))
colnames(combinat)<-c('Actual','Fitted')
# plot
Timeline<-seq(as.Date("2011-02-01"), as.Date("2012-12-24"), by='days')
best_n <- data.frame(Timeline, combinat)
best_n = xts(x=best_n[,-1], order.by=best_n$Timeline)
dygraph(contr_pl) %>% dyRangeSelector()
```



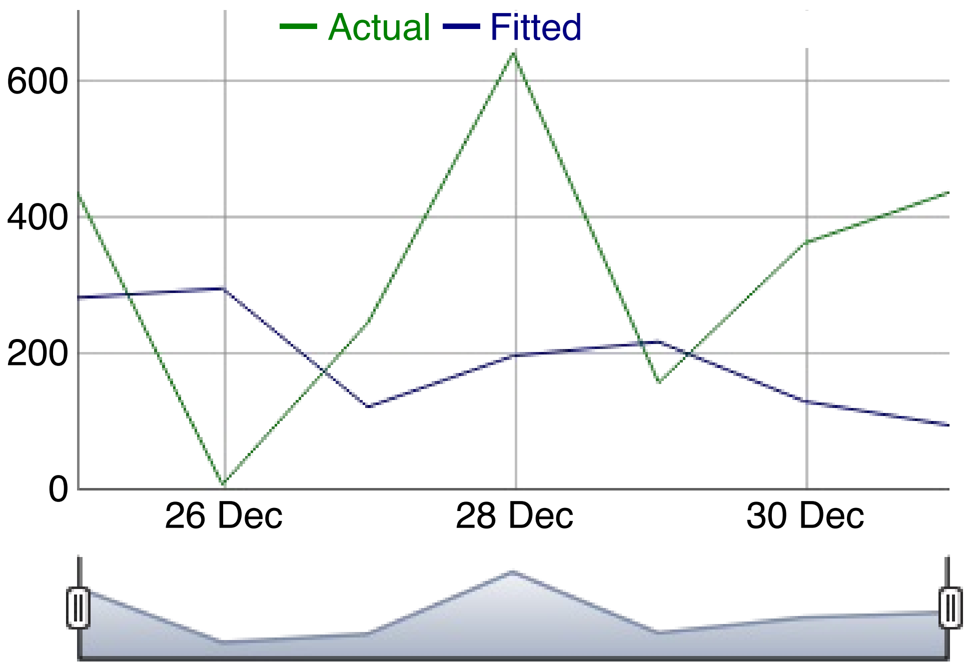
As we can notice the model is almost perfect to be much better than the other models we have seen from above. The outcome and performance of the model is extensively better.

Prediction intervals for NNETAR models

```
# forecast 7 points
fit.nn0 = accuracy(forecast(mod.nn, h=7), tm.test)
kable(fit.nn0)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-0.9931704	92.09217	56.82937	-11.17487	17.44446	0.1438924	-0.0088996
Test set	136.1452191	266.46822	235.73970	-420.86630	505.71258	0.5968945	NA

```
# prediction plot
fcast <- forecast(mod.nn, PI=TRUE, h=7)
ft<-as.data.frame(fcast)[,1]
best_npt<-cbind(as.vector(tm.test), as.vector(ft))
colnames(best_npt)<-c('Actual','Fitted')
# plot
Timeline<-seq(as.Date("2012-12-25"), as.Date("2012-12-31"), by='days')
best_npt <- data.frame(Timeline, best_npt)
best_npt = xts(x=best_npt[,-1], order.by=best_npt$Timeline)
dygraph(best_npt) %>% dyRangeSelector()
```



The model can be written as

$$y_t = f(\boldsymbol{y}_{t-1}) + \epsilon_t,\ \epsilon_t \sim N(0,1)$$

where $\boldsymbol{y}_{t-1}$ is a vector containing lagged values of the series, and f is a neural network with hidden nodes in a single layer.

However, for small prediction points, 7 days, the neural network model has some discrepancy to be large enough. In this cases, we could also come up with simulations on different model to construct for better forecast on the last week of 2012.

- **Reference:** https://www.displayr.com/how-to-add-trend-lines-in-r-using-plotly/ (estimation for visual inspection)
- **Reference:** https://rstudio.github.io/dygraphs/ (Dynamic Graph)
- **Reference:** https://robjhyndman.com/hyndsight/arma-roots/ (Unit Root test)
- **Reference:** William W.S. Wei (2006), *Time Series Analysis–Univariate and Multivariate Methods*, Second Edition. (Chapter 14)
- **Reference:** https://robjhyndman.com/hyndsight/nnetar-prediction-intervals/ (Time series modeling, Neural network)
- **Reference:** Shumway, R. H., & Stoffer, D. S. (2017). Time series analysis and its applications: with R examples. 4th edition (p.265-270, ch.5, p.396, ch.7).
- **Reference:** University of Toronto, STA 457, Ljung Box test codes, Prewhitening process taught/codes provided in 2020 Fall term by Kyuson.Lim.