BANA 7050 Time series forecasting - Final project

Forecasting hourly new users for a mobile app

Utkarsh Singh (M13457333)

03/12/2019

INTRODUCTION

Number of new users per hour is an important metric to measure customer engagement and acquisition for mobile apps. Amongst all other factors, current usage (which contributes to app popularity) and app performance can impact new user turnout in the future. This report analyses hourly recorded data on active users, total number of active sessions and app crashes.

OBJECTIVE

Objective of this report is to analyse the impact of total users, total active sessions and number of crashes in the current hour on the number of new users in the next hour. A vector error correction model is used to generate forecasts for the next 48 hours.

DATA

The data for this report has been downloaded from the following source -> https://www.kaggle.com/wolfgangb33r/usercount

The dataset contains four time series data of an hourly resolution for a period of 1 week. The following series are reported:

- **Users**: Total number of active users
- **Sessions**: Total number of active sessions
- **Crashes**: Total number of app crashes reported
- **New users**: Total number of new users

FINDINGS

Analysis suggests that an increase in app crashes leads to a significant reduction in new users. The estimate for the "*crashes*" variable in the VEC model is found to be **-0.91**

```r
head(appdata,5)
```

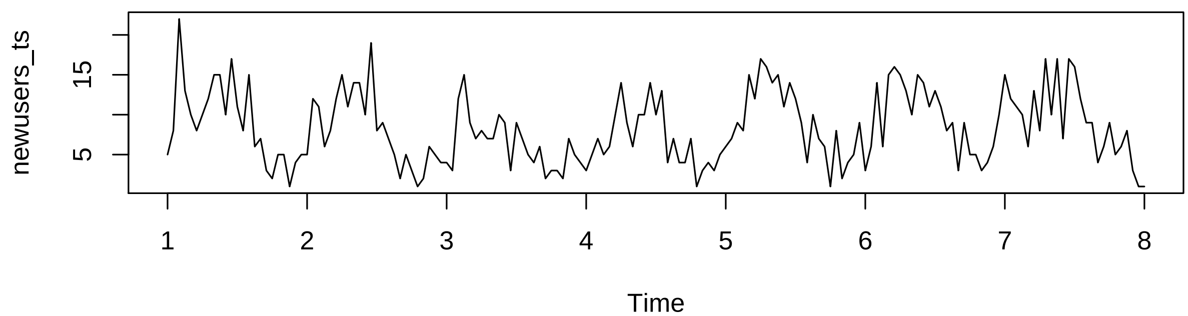
```
##             time users sessions newusers crashes
## 1 22.12.18 09:00    64       60        5       0
## 2 22.12.18 10:00    79       84        8       0
## 3 22.12.18 11:00    97      102       22       0
## 4 22.12.18 12:00   107      102       13       0
## 5 22.12.18 13:00   105      117       10       2
```

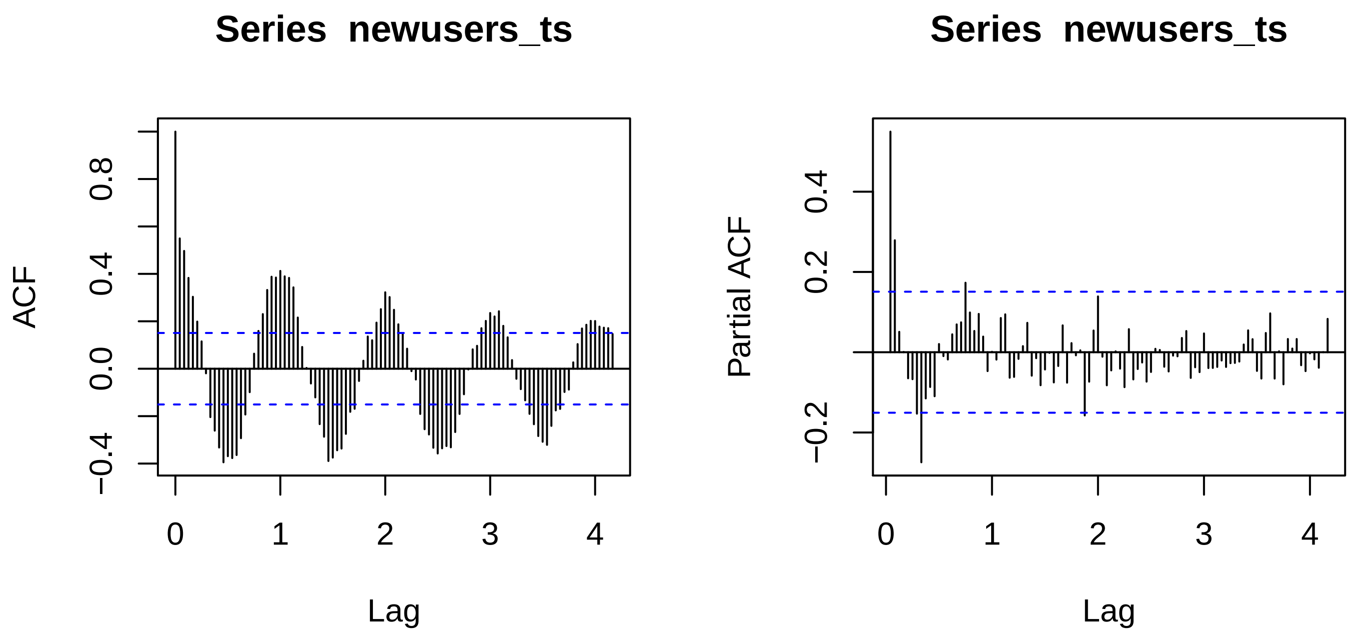
Part 1: Seasonal Adjustment

This section identifies and adjusts for seasonality in the data. Since the data is recorded hourly, there is a high possibility that the app usage behavior is periodic in nature. First seasonality is studied for the target variable, and the measures to adjust seasonality are extended to other variables.

Plot hourly new users data

```r
library(tseries)
library(forecast)
newusers <- appdata[,c("newusers")]
newusers_ts <- ts(newusers,start=1,frequency = 24)
plot.ts(newusers_ts)
```





The ACF plot shows periodic spikes in the data, which is indicative of seasonality. Data is tested for a unit root using ADF tests.

Test for stationarity and seasonality

ADF Test

```
library(tseries)
library(forecast)

(adftest <- adf.test(newusers_ts, k=24))
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  newusers_ts
## Dickey-Fuller = -1.6535, Lag order = 24, p-value = 0.7209
## alternative hypothesis: stationary
```

At a lag order of 24, the ADF test shows that a unit root exists and hence data is non-stationary.Data was further tested for stationarity assuming “trend”, “lag” or “none”.

```
library(urca)

summary(ur.df(newusers_ts,type="none"))
summary(ur.df(newusers_ts,type="drift"))
summary(ur.df(newusers_ts,type="trend"))
```

type	test.stat	CV_1pct	CV_5pct	CV_10pct	R_sq
none	-1.908	-2.58	-1.95	-1.62	0.2087
drift	-4.29	-3.46	-2.88	-2.57	0.273
trend	-4.26	-3.99	-3.43	-3.13	0.267

The results show that there is no significant trend or drift component in the data. But the null hypothesis of a unit root is not rejected at 5% α level.

The *tbats* function allows a seasonal component in the model and can be used to report if any significant seasonal component is detected.

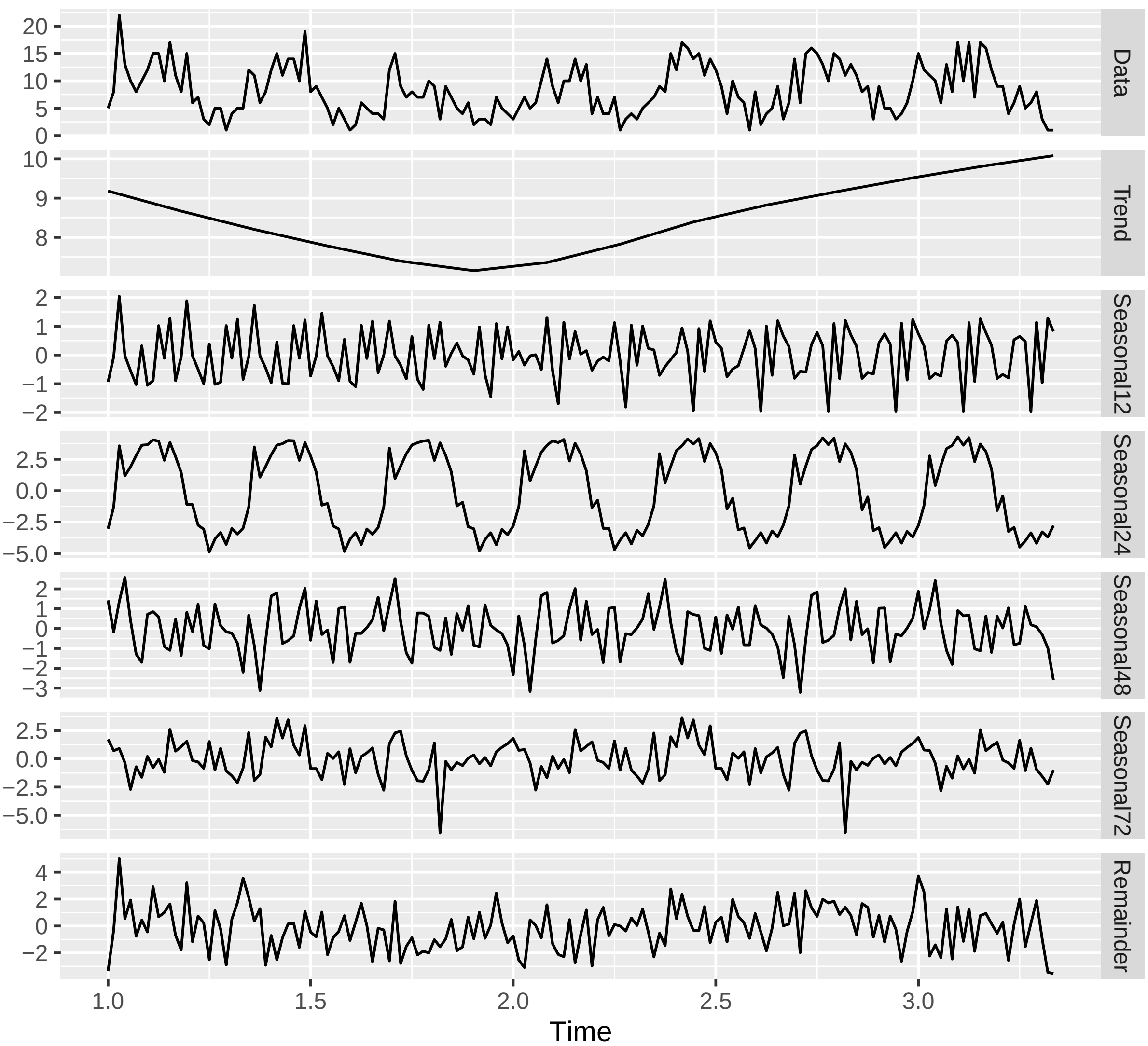
```
fit <- tbats(newusers_ts)
(seasonal <- !is.null(fit$seasonal))
```

```
## [1] TRUE
```

The TBATS model confirms seasonality. Also, since the data is captured at an hourly level, there's a possibility of **multiple seasonalities**

Decompose and adjust for seasonality

```
newusers.decomp <- newusers %>% msts(seasonal.periods = c(12,24,48,72)) %>% mstl()
autoplot(newusers.decomp)
```



Multiple seasonalities are observed in the data. These components are subtracted from the data to adjust for seasonality. A first order difference is also taken to check for stationarity.

```
newusers.seasadj1 <- newusers.decomp[, "Data"] - (newusers.decomp[,
    "Seasonal12"] + newusers.decomp[, "Seasonal24"] + newusers.decomp[,
    "Seasonal48"] + newusers.decomp[, "Seasonal72"])

newusers.seasadj2 <- newusers.seasadj1 %>% diff()

summary(ur.df(newusers.seasadj1, type = "none"))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
```

```
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1079 -1.1710  0.0666  1.3026  6.9186
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.01730    0.01711  -1.012    0.313
## z.diff.lag -0.49220    0.06722  -7.322 1.02e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.883 on 165 degrees of freedom
## Multiple R-squared:  0.2584, Adjusted R-squared:  0.2494
## F-statistic: 28.75 on 2 and 165 DF,  p-value: 1.939e-11
##
##
## Value of test-statistic is: -1.0116
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```r
summary(ur.df(newusers.seasadj2, type = "none"))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.6428 -1.2431 -0.0421  1.0501  4.5000
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -2.03972    0.12164 -16.768  < 2e-16 ***
## z.diff.lag  0.34037    0.07003   4.861 2.72e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.699 on 164 degrees of freedom
## Multiple R-squared:  0.801,  Adjusted R-squared:  0.7986
## F-statistic: 330.1 on 2 and 164 DF,  p-value: < 2.2e-16
```

```
##
##
## Value of test-statistic is: -16.7684
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

The above test results show that the newusers data is not stationary after seasonal adjustment but is stationary after first-order differencing.

```r
fit <- tbats(newusers.seasadj1)
(seasonal <- !is.null(fit$seasonal))
```

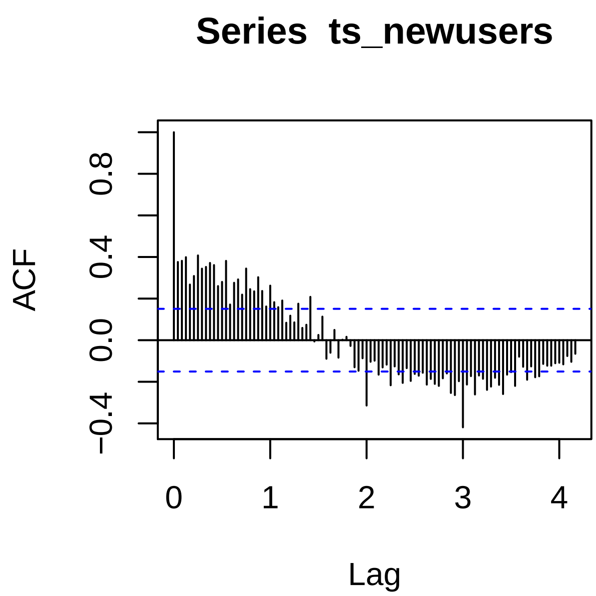
```
## [1] FALSE
```

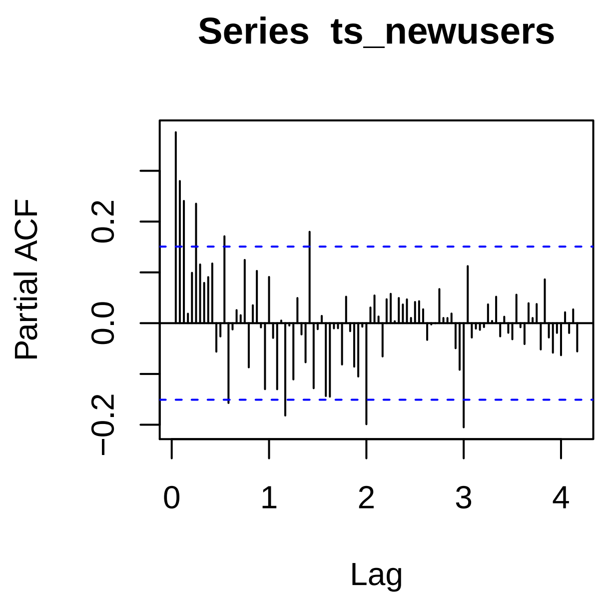
The following observations are made for the series "newusers" so far:

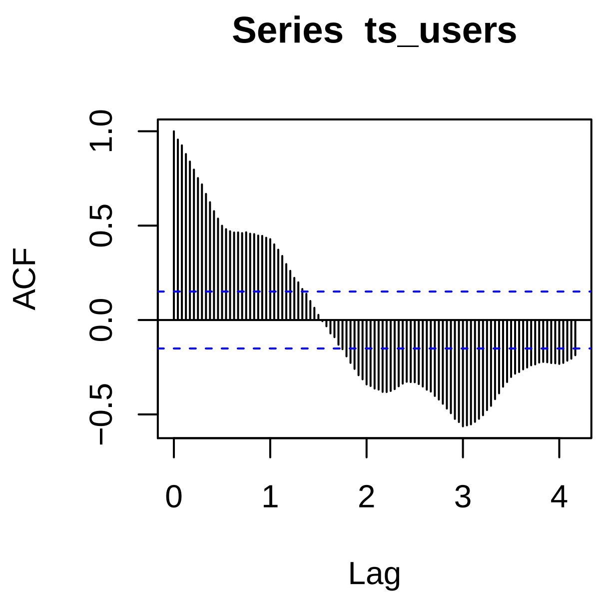
- The series is seasonally adjusted but non-stationary
- The series becomes stationary upon 1st order differencing. Therefore "newusers" is I(1)

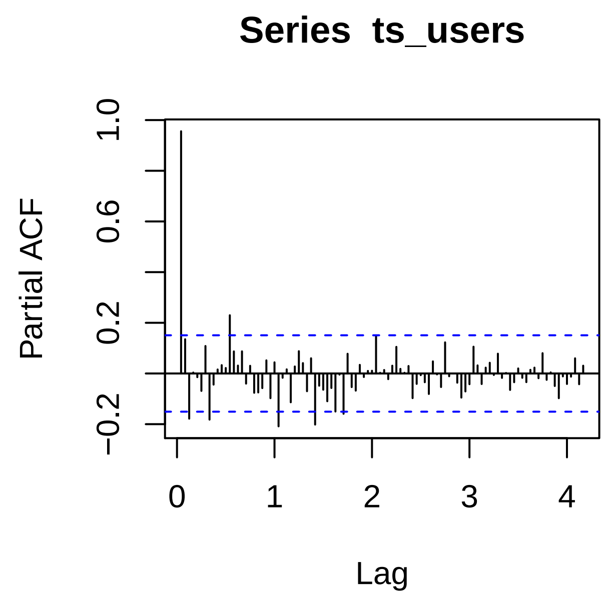
Similarly, the rest of the series in data were adjusted for seasonality. The final plots of the time series present in data are shown below. Please note the series after seasonal adjustment are non-stationary. Stationarity is achieved on first-order differencing

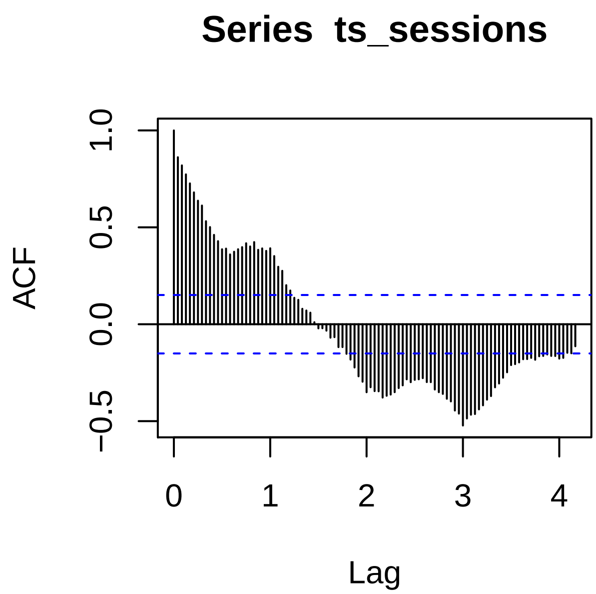
```r
ts_users <- ts(seasadj_appusers_data1$users.seasadj1, start = 1, frequency = 24)
ts_sessions <- ts(seasadj_appusers_data1$sessions.seasadj1, start = 1, frequency = 24)
ts_newusers <- ts(seasadj_appusers_data1$newusers.seasadj1, start = 1, frequency = 24)
ts_crashes <- ts(seasadj_appusers_data1$crashes.seasadj1, start = 1, frequency = 24)
```

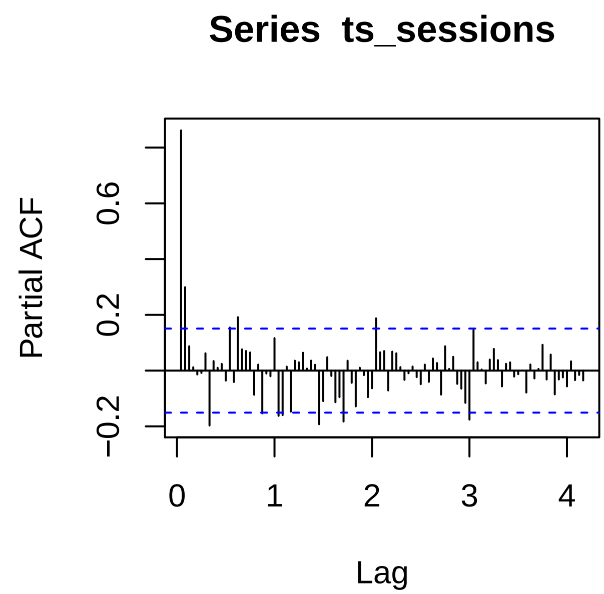


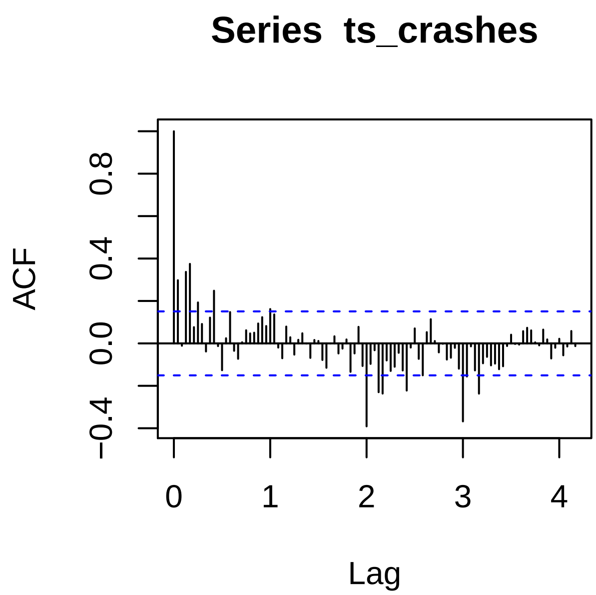


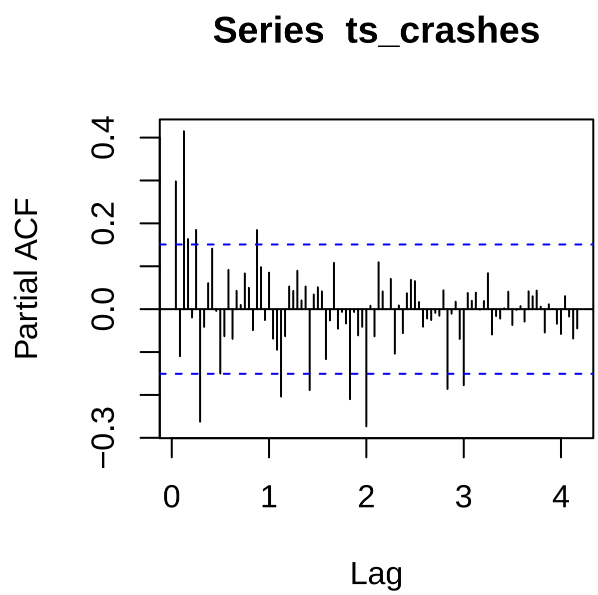

Series ts_users
ACF
−0.5
0.0
0.5
1.0
0
1
2
3
4
Lag


Series ts_users
Partial ACF
−0.2
0.2
0.6
1.0
0
1
2
3
4
Lag


Series ts_sessions
ACF
−0.5
0.0
0.5
1.0
0
1
2
3
4
Lag


Series ts_sessions
Partial ACF
−0.2
0.2
0.6
0
1
2
3
4
Lag





Notes

- The multiple time series in the data are seasonally adjusted. Some residual seasonal component might still remain due to non-integral periods
- The time series achieve stationarity after first-order differencing, therefore all series are I(1)

Part 2: Cointegration

Two series Z_t and Y_t, each of integrated order (1), are said to be cointegrated if they have a same or common stochastic trend that can be eliminated by taking a specific difference of the series such that the resultant series is stationary. To perform conintegration tests on the data, the "**Phillips-Ouliaris**" test is conducted using the R function po.test.

Test for cointegration

```
mat1 <- as.matrix(cbind(ts_newusers,ts_users,ts_sessions,ts_crashes), demean=FALSE)
po.test(mat1)
```

```
## Warning in po.test(mat1): p-value smaller than printed p-value
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  mat1
## Phillips-Ouliaris demeaned = -173.11, Truncation lag parameter =
## 1, p-value = 0.01
```

test shows cointegration exists between the series

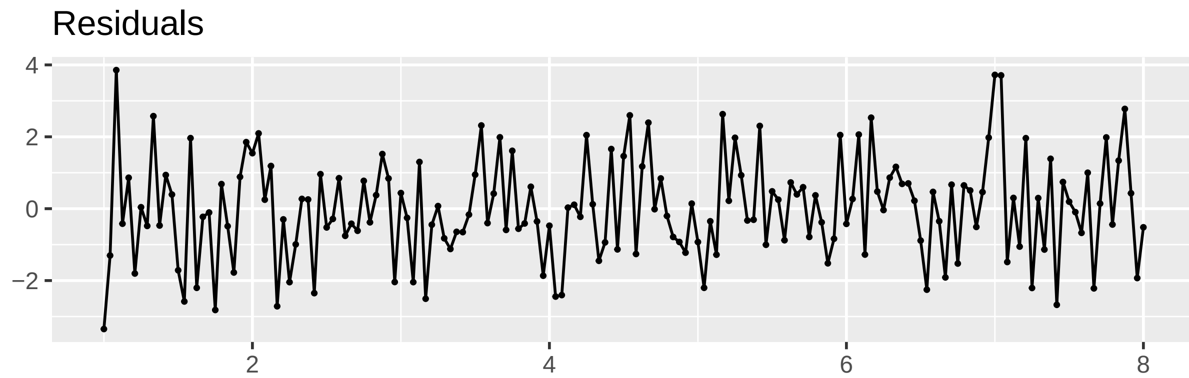
```r
library("dynlm")
reg.dyn <- dynlm(ts_newusers~ ts_users + ts_sessions + ts_crashes)
ehat <- resid(reg.dyn)
(adf.test(ehat))
```

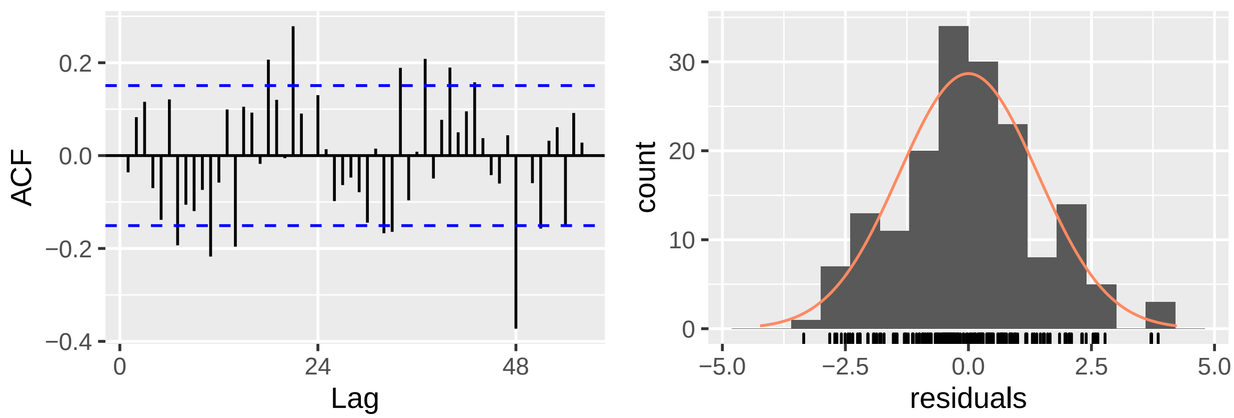
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ehat
## Dickey-Fuller = -5.2515, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

```r
summary(ur.df(ehat,type="none",lags = 1))
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8052 -0.8433 -0.0671  0.7416  4.0678
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.96564    0.11095  -8.703 3.18e-15 ***
## z.diff.lag -0.08190    0.07649  -1.071    0.286
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.39 on 165 degrees of freedom
## Multiple R-squared:  0.5298, Adjusted R-squared:  0.5241
## F-statistic: 92.96 on 2 and 165 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -8.7032
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```r
checkresiduals(ehat)
```





In conclusion, the null hypothesis that the residuals have unit roots is rejected, and therefore the series are cointegrated.

We have earlier seen that the series have an order of integration = 1.

With cointegrated series we can construct a VEC model to better understand the causal relationship between the two variables.

```
summary(reg.dyn)
```

```
##
## Time series regression with "ts" data:
## Start = 1(1), End = 8(1)
##
## Call:
## dynlm(formula = ts_newusers ~ ts_users + ts_sessions + ts_crashes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3475 -0.8784 -0.0380  0.8415  3.8556
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.49852    0.57875   2.589  0.01048 *
## ts_users     0.02979    0.02010   1.482  0.14022
## ts_sessions  0.06867    0.02340   2.935  0.00381 **
## ts_crashes  -0.91520    0.29652  -3.087  0.00238 **
```

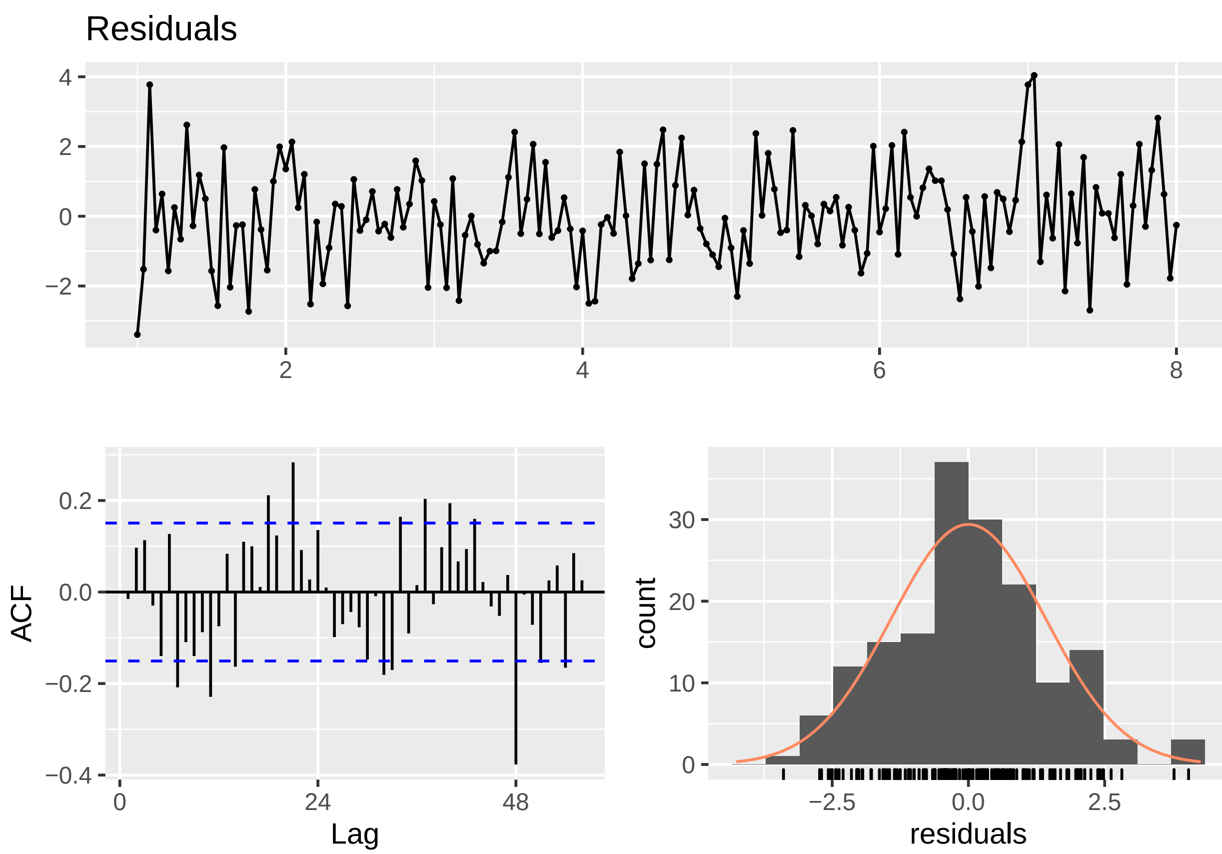
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.424 on 165 degrees of freedom
## Multiple R-squared:  0.4827, Adjusted R-squared:  0.4733
## F-statistic: 51.33 on 3 and 165 DF,  p-value: < 2.2e-16
```

Note:

- the negative sign in the coefficient of "crashes", which indicates the opposite relationship between the number of new users and the number of times the app has crashed in the previous hour
- The coefficient for "*users*" is not significant, hence it is removed from the model

```r
library("dynlm")
reg.dyn_2 <- dynlm(ts_newusers~  ts_sessions + ts_crashes)
ehat_2 <- resid(reg.dyn_2)

checkresiduals(ehat_2)
```




```r
Box.test(resid(reg.dyn_2),type = "Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid(reg.dyn_2)
## X-squared = 0.039426, df = 1, p-value = 0.8426
```

The residuals are a white noise process, hence the fitted model is adequate

```r
vecmodel <- dynlm(d(ts_newusers)~L(ehat_2))

summary(vecmodel)
```

```
##
## Time series regression with "ts" data:
## Start = 1(2), End = 8(1)
##
## Call:
## dynlm(formula = d(ts_newusers) ~ L(ehat_2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5391 -1.2545 -0.0885  1.0377  4.8735
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.005832   0.125967   0.046    0.963
## L(ehat_2)   -1.023666   0.088680 -11.543   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.633 on 166 degrees of freedom
## Multiple R-squared:  0.4453, Adjusted R-squared:  0.4419
## F-statistic: 133.3 on 1 and 166 DF,  p-value: < 2.2e-16
```

The coefficient for the error correction term L(ehat_2) is significant, suggesting that changes in total number of sessions and crashes in an hour does affect the number of new users in the next hour.

Part 3: Forecasting using VEC model

Select number of lags using VARselect

Number of lags to be used in the model is identified using the differenced series (shown above as stationary).

```r
library(vars)
dy = cbind(users.seasadj2,newusers.seasadj2,sessions.seasadj2,crashes.seasadj2)
VARselect(dy,lag.max=12, type="const")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##     12     12      2     12
##
## $criteria
##                  1          2          3         4         5         6
## AIC(n)    5.501322   4.646998   4.628258  4.424045  4.211376  3.647280
## HQ(n)     5.660132   4.932856   5.041164  4.964000  4.878379  4.441331
## SC(n)     5.892329   5.350811   5.644876  5.753470  5.853606  5.602316
## FPE(n) 245.037048 104.325015 102.494314 83.723627 67.894366 38.800270
```

```
##                7         8         9        10        11        12
## AIC(n)  3.611125  3.379927  3.014908  3.023000  2.621937  2.278928
## HQ(n)   4.532224  4.428074  4.190104  4.325243  4.051229  3.835267
## SC(n)   5.878967  5.960574  5.908361  6.229259  6.141002  6.110798
## FPE(n) 37.660842 30.140498 21.152096 21.617906 14.722013 10.663553
```

The BIC indicates 2 lags, but an extra lag is added as BIC tends to under-parametrize.

```r
y <- cbind(ts_newusers,ts_users,ts_sessions,ts_crashes)
vecm1= ca.jo(y,ecdet="const",type="eigen",K=3,spec="longrun", season = 24)
summary(vecm1)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: maximal eigenvalue statistic (lambda max) , without linear trend and constant in cointegra
##
## Eigenvalues (lambda):
## [1] 2.629139e-01 2.373242e-01 1.307052e-01 3.102969e-02 1.665335e-16
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 3 |  5.23  7.52  9.24 12.97
## r <= 2 | 23.25 13.75 15.67 20.20
## r <= 1 | 44.97 19.77 22.00 26.81
## r = 0  | 50.64 25.56 28.14 33.24
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                ts_newusers.l3 ts_users.l3 ts_sessions.l3 ts_crashes.l3
## ts_newusers.l3    1.000000000  1.00000000       1.000000     1.0000000
## ts_users.l3      -0.002205529 -0.06547213      -1.068487     0.4137151
## ts_sessions.l3   -0.160642064  0.02508286       1.119390    -0.1685519
## ts_crashes.l3     4.648338916 -2.77200119       7.727586     4.0708605
## constant          1.778185370 -4.35376040     -11.231690   -26.8859307
##                    constant
## ts_newusers.l3   1.00000000
## ts_users.l3     -0.07390824
## ts_sessions.l3   0.19426699
## ts_crashes.l3    3.83837071
## constant       -32.25106471
##
## Weights W:
## (This is the loading matrix)
##
##               ts_newusers.l3 ts_users.l3 ts_sessions.l3 ts_crashes.l3
## ts_newusers.d     -0.1365712 -0.51799553   -0.002758811 -0.0268436567
## ts_users.d         0.1269646  0.38904664   -0.110663219 -0.1100101245
## ts_sessions.d      0.2508416 -0.01761248   -0.420921230 -0.1029329966
```

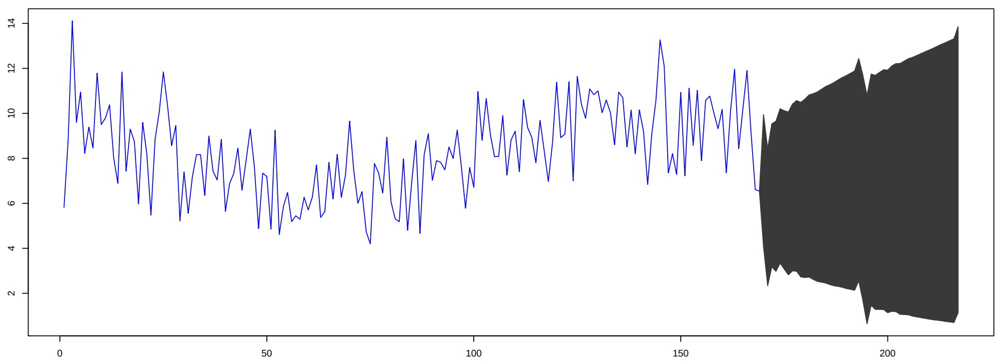
```
## ts_crashes.d      -0.1438319  0.06805088   -0.003281797 -0.0001527213
##                    constant
## ts_newusers.d  4.606183e-17
## ts_users.d     3.548655e-16
## ts_sessions.d  1.751027e-16
## ts_crashes.d  -3.235164e-17
```

The Johansen test confirms that 2 cointegrations exist (It has already been established that the series in our data are cointegrated). A linear combination of 2 time series is required to form a stationary series.

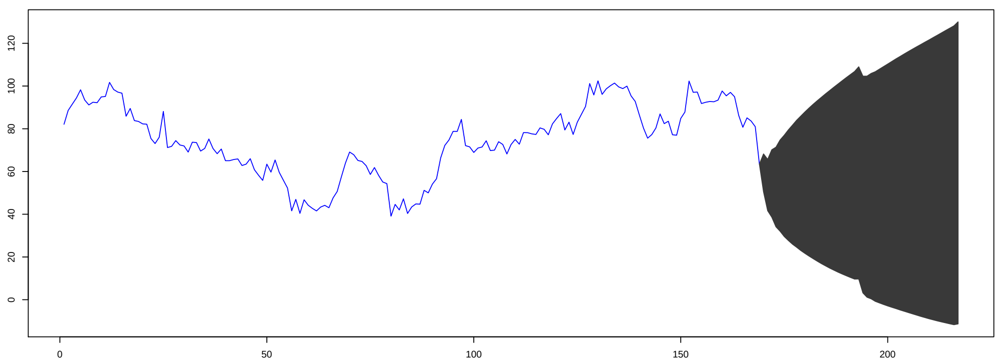
Forecast

```r
library( vars)
varf=vec2var(vecm1,r=2) # r=2 means USE 2 LR relationship test results
fcast= predict(varf,n.ahead=48,ci=0.95)
fanchart(fcast, cis = 0.95, col.y = "blue")
```

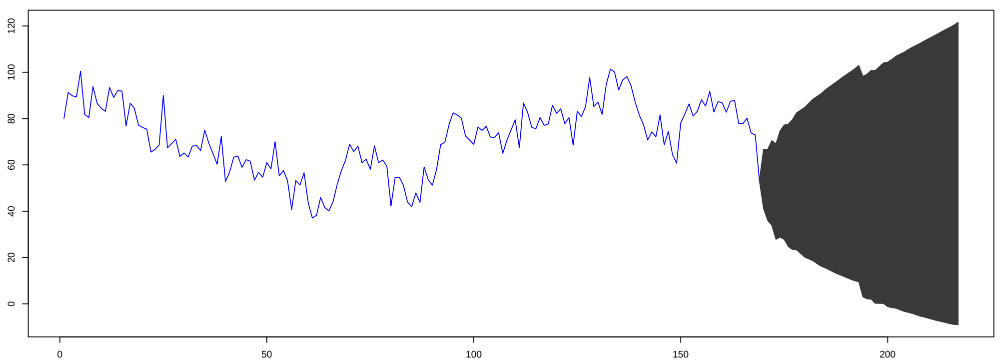
Fanchart for variable ts_newusers



Fanchart for variable ts_users



Fanchart for variable ts_sessions



Fanchart for variable ts_crashes