Configuration Manual

MSc Research Project
FinTech

Sneha Sadaye
Student ID: x19107692

School of Computing
National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sneha Sadaye
Student ID:	X19107692
Programme:	MSc in FinTech
Year:	2020
Module:	MSc Research Project
Lecturer:	Noel Cosgrave
Submission Due Date:	17th August 2020
Project Title:	Analysing the Impact of Demonetisation on Digital Payments in India
Word Count:	4685
Page Count:	18

Signature: ……………………………………………………………………………………………………

Date: 17th August 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Sneha Sadaye
Student ID: x19107692

1 Introduction

This document describes the hardware and software requirements to perform the research for the thesis titled "*Analysing the Impact of Demonetisation on Digital Payments in India*". It also lays out the steps to be followed to replicate the research and perform all the experiments in a precise manner to ensure that the research can be easily reproduced, and results can be analysed. To ensure in depth explanation of the steps performed, code snippets and important outputs are also portrayed.

2 Data Collection

The data for this research is collected from Reserve Bank of India's (RBI) Database on Indian Economy[1]. It includes the monthly value and volume of financial transactions performed through various payment systems in the period of April 2004 to October 2019. The data is available in CSV format.

3 Hardware Configuration

The configuration of the laptop used for performing the research is presented in Table 1.

Table 1: Device configuration

Processor	Intel® Core™ i5-5200U CPU @ 2.20GHz
Installed RAM	8.00 GB
System type	64-bit operating system, x64-based processor
Operating System Edition	Windows 10 Pro

4 Software Configuration

The software tools used are shown in Table 2.

Table 2: Software configuration

Programming language	R version 3.6.1
Integrated Development Environment (IDE)	RStudio version 1.2.1335

[1] https://dbie.rbi.org.in/DBIE/dbie.rbi?site=home

5 Steps in the Analysis

The steps performed in the research are described in detail in the following sections.

5.1 Importing libraries

In RStudio, the relevant packages are installed and libraries are imported to use the time series analysis functions and various other functions used for processing and visualising the data. These libraries include forecast, ggplot2, tseries, lmtest, strucchange, tidyr, tsoutliers, TSA and nlme.

5.2 Loading Data

The following code loads the data in RStudio:

```
pay_data <-  read.csv("RBIB_Table_No._43___Payment_System_Indicators.csv",
stringsAsFactors = F)
```

5.3 Pre-processing Data

The data contains some rows having descriptions of the data, for example, the beginning of each year etc. Such rows are removed from the data. Also, the first two rows containing headers are removed. The code to do this is as follows:

```
pay_data_new <- pay_data[c(-3, -11, -24, -37, -50, -63, -76, -89, -102, -115, -128,
-141, -154, -167, -180, -193, -206, -207, -208),]
pay_data_new <- pay_data_new[c(-1,-2),]
```

The operations such as changing date to the desired format, converting datatype of numeric values and ordering the data according to date are done using the following code:

```
# Change Date format to YYYY-MM-DD
pay_data_new$Date <- as.Date(paste("01-", pay_data_new$Date, sep = ""), format =
"%d-%b-%Y")

# Order by Date
pay_data_new <- pay_data_new[order(pay_data_new$Date), ]

# Convert to numeric
pay_data_new[c(2:length(pay_data_new))] <-
as.data.frame(sapply(pay_data_new[,c(2:length(pay_data_new))], as.numeric))
```

The raw data contains columns that are not useful for this research. These columns are removed and a new dataframe is created with useful columns that represent transactions through digital payment modes.

```
#Select the variables of interest (only digital modes of payment)
pay_data_digi <- pay_data_new[, c(1, 2, 3, 36, 37, 54, 55, 60, 61, 64, 65, 70, 71)]
colnames(pay_data_digi) <- c("Date", "RTGS Volume", "RTGS Value", "REC Volume",
"REC Value","CC POS Volume", "CC POS Value", "DC POS VOlume", "DC POS Value","m-
Wallet Volume", "m-Wallet Value", "Mobile Banking Volume", "Mobile Banking Value")
```

5.4 Exploratory Data Analysis

This step includes separating the individual payment system transactions as individual datasets in order to visualise these over the period of study. These data are plot using the following code:

```
# By Volume
ggplot() +
geom_line(data = rtgs_data_vol, aes(x = Date, y = Volume, color = "RTGS")) +
geom_line(data = rec_vol, aes(x = Date, y = Volume, color = "Retail Electronic
Clearing")) +
geom_line(data = cardPOS_vol, aes(x = Date, y = Volume, color = "Total Card Usage
at POS")) +
geom_line(data = mwallet_vol, aes(x = Date, y = Volume, color = "m-Wallet")) +
geom_line(data = mobbank_vol, aes(x = Date, y = Volume, color = "Mobile Banking"))
+
scale_color_manual("",
breaks = c("RTGS", "Retail Electronic Clearing", "Total Card Usage at POS",
"m-Wallet", "Mobile Banking"),
values = c('#FFC300','#00AA11', '#F955BB',
"#9D75D7", '#00AFBC')) +
labs(title = "Digital Payments: Mode-wise Transaction Volume",
x = "Year",
y = "Lakhs") +
scale_x_date(date_labels = "%Y", date_breaks = "1 year") +
geom_vline(xintercept = as.Date("08-11-2016",  format = "%d-%m-%Y"), color =
'#FF0000', linetype="dotted")  +
theme(axis.text.x = element_text(face = "bold", size = 5, angle = 45))

# By Value
ggplot() +
geom_line(data = rtgs_data_value, aes(x = Date, y = Value/1000, color = "RTGS")) +
geom_line(data = rec_value, aes(x = Date, y = Value/1000, color = "Retail
Electronic Clearing")) +
geom_line(data = cardPOS_value, aes(x = Date, y = Value/1000, color = "Total Card
Usage at POS")) +
geom_line(data = mwallet_value, aes(x = Date, y = Value/1000, color = "m-Wallet"))
+
geom_line(data = mobbank_value, aes(x = Date, y = Value/1000, color = "Mobile
Banking")) +
scale_color_manual("",
breaks = c("RTGS", "Retail Electronic Clearing", "Total Card Usage at POS",
"m-Wallet", "Mobile Banking"),
values = c('#FFC300','#00AA11', '#F955BB',
"#9D75D7", '#00AFBC')) +
labs(title = "Digital Payments: Mode-wise Transaction Value",
x = "Year",
y = "10 Billion INR") +
scale_x_date(date_labels = "%Y", date_breaks = "1 year") +
geom_vline(xintercept = as.Date("08-11-2016",  format = "%d-%m-%Y"), color =
'#FF0000', linetype="dotted")  +
theme(axis.text.x = element_text(face = "bold", size = 5, angle = 45))
```

This step also includes calculating transactions in each payment mode as percentage of the total transactions (total of cash-based and digital). This is done in order to determine the change in trend of each payment mode as a percentage of overall trend. The code to calculate the percentage and plot the data is given below:

```
# 1. RTGS value as percentage of total value
rtgs_val_perc <- rtgs_data_value$Value/pay_data_all$total_value *100
rtgs_val_perc_df <- as.data.frame(rtgs_data_value$Date)
rtgs_val_perc_df <- cbind(rtgs_val_perc_df,rtgs_val_perc)
colnames(rtgs_val_perc_df) <- c("Date", "Percentage")
```

```
# 2. REC value as percentage of total value
rec_val_perc <- rec_value$Value/pay_data_all$total_value *100
rec_val_perc_df <- as.data.frame(rec_value$Date)
rec_val_perc_df <- cbind(rec_val_perc_df,rec_val_perc)
colnames(rec_val_perc_df) <- c("Date", "Percentage")

# 3. Card usage at POS value as percentage of total value
card_val_perc <- cardPOS_value$Value/pay_data_all$total_value *100
card_val_perc_df <- as.data.frame(cardPOS_value$Date)
card_val_perc_df <- cbind(card_val_perc_df,card_val_perc)
colnames(card_val_perc_df) <- c("Date", "Percentage")

#4. m-Wallet value as percentage of total value
length(mwallet_value$Value)
mwallet_val_perc <- mwallet_value$Value/pay_data_all$total_value[c(97:187)] *100
mwallet_val_perc_df <- as.data.frame(mwallet_value$Date)
mwallet_val_perc_df <- cbind(mwallet_val_perc_df, mwallet_val_perc)
colnames(mwallet_val_perc_df) <- c("Date", "Percentage")

#5. Mobile Banking value as percentage of total value
length(mobbank_value$Value)
length(pay_data_all$total_vol)
mobbank_val_perc <- mobbank_value$Value/pay_data_all$total_value[c(85:187)] *100
mobbank_val_perc_df <- as.data.frame(mobbank_value$Date)
mobbank_val_perc_df <- cbind(mobbank_val_perc_df, mobbank_val_perc)
colnames(mobbank_val_perc_df) <- c("Date", "Percentage")


#Plot - By Value
ggplot() +
  geom_line(data = rtgs_val_perc_df, aes(x = Date, y = Percentage/10, color =
"RTGS")) +
  geom_line(data = rec_val_perc_df, aes(x = Date, y = Percentage, color = "Retail
Electronic Clearing")) +
  geom_line(data = card_val_perc_df, aes(x = Date, y = Percentage, color = "Total
Card Usage at POS")) +
  geom_line(data = mwallet_val_perc_df, aes(x = Date, y = Percentage, color = "m-
Wallet")) +
  geom_line(data = mobbank_val_perc_df, aes(x = Date, y = Percentage, color =
"Mobile Banking")) +
  scale_color_manual("",
                     breaks = c("RTGS", "Retail Electronic Clearing", "Total Card
Usage at POS","m-Wallet", "Mobile Banking"),
                     values = c('#FFC300','#00AA11', '#F955BB',
                                "#9D75D7", '#00AFBC')) +
  labs(title = "Mode-wise Transaction Value % of Total Transaction Value",
       x = "Year") +
  scale_x_date(date_labels = "%Y", date_breaks = "1 year") +
  scale_y_continuous(name = "Percentage",
                     sec.axis = sec_axis(~.*10, name = "RTGS Percentage"))+
  geom_vline(xintercept = as.Date("08-11-2016",  format = "%d-%m-%Y"), color =
'#000000', linetype = "dotted")  +
  theme(axis.text.x = element_text( size = 8, angle = 45))
```

The above code is for the value data. A similar EDA for volume data is also performed.

For the analysis performed in this study, a sum total of all the digital transactions in terms of volume and value is used. This represents the transactions performed through all the five modes described above as single dataset. The sum of digital transactions and its percentage of the total transactions is calculated as shown below:

```
## Calculate sum of all digital modes
pay_data_digi$`Total Volume` <- rowSums(pay_data_digi[,c(2,4,6,8,10)], na.rm = T)
pay_data_digi$`Total Value` <- rowSums(pay_data_digi[,c(3,5,7,9,11)], na.rm = T)

# Digital transaction volume as percentage of total volume
```

```
digital <- pay_data_digi[,c(1,12,13)]
colnames(digital) <- c('Date', 'Volume', 'Value')

# Remove the first 12 months as they show almost 100%
digital_volume_perc <- digital$Volume[-c(1:12)]/pay_data_all$total_vol[-c(1:12)]
*100

#Create a dataframe
digital_volume_perc_df <- as.data.frame(digital$Date[-c(1:12)])
digital_volume_perc_df <- cbind(digital_volume_perc_df, digital_volume_perc)
colnames(digital_volume_perc_df) <- c("Date", "Percentage")
#Plot
ggplot(digital_volume_perc_df, aes(x = Date, y = Percentage)) +
  geom_line(color = "#00AFBC", size = 1) +
  labs(title = "Digital Transaction Volume as Percentage of Total Volume",
       x = "Month-Year", y = "Percentage") +
  scale_x_date(date_labels = "%m-%y", date_breaks = "1 year") +
  geom_vline(xintercept = as.Date("08-11-2016",  format = "%d-%m-%Y"), color =
'red')
```

5.5 Preliminary analysis using Segmented Regression

For determining the change caused by demonetisation, the data of all digital transactions is summed and calculated as a percentage of total. The data is then analysed using segmented regression. The date of demonetisation is used to create variables to represent intervention. A variable for time encoding T is created, starting with 1 and increasing successively by 1 for all time points. The variable for intervention X is coded as 0 before demonetisation and 1 after demonetisation. The variable Ti is coded as 0 before demonetisation and incremented by 1 successively after demonetisation. These variables are used to fit regression lines to the pre- and post-demonetisation data. The gls() function is used. To account for the autocorrelation in data, an Autoregressive and Moving Average order is set using the corARMA() function and passed to the correlation parameter. The code for segmented regression for digital volume data is as follows:

```
digi_vol_lm_df <- digital_volume_perc_df
digi_vol_lm_df$T = as.numeric(1:length(digi_vol_lm_df$Percentage))

#Define xreg = 0 pre-intervention and 1 post-intervention
pre_inter <- digi_vol_lm_df$Date < as.Date("2016-11-01")
ind_pre <- which(pre_inter)
ind_post <- which(!pre_inter)
xreg = rep(0,length(digi_vol_lm_df$Percentage))
for (i in ind_post) {
  xreg[i] = 1
}
digi_vol_lm_df$X <- xreg

Ti = rep(0,length(digi_vol_lm_df$Percentage))
for (i in ind_post) {
  Ti[i] = Ti[i-1] + 1
}
digi_vol_lm_df$'Ti' <- Ti
# Fit the regression model
gls.model.1 <- gls(Percentage ~ T + X + Ti,
                   data = digi_vol_lm_df,
                   correlation = corARMA(form = ~ T, p = 1, q = 1),
                   method = "ML")
checkresiduals(gls.model.1)
```

The residuals of this model suggest that autocorrelation is present and that the residuals are not completely white noise. The QQ plot and ACF plot of the model residuals are shown in Figure 1.

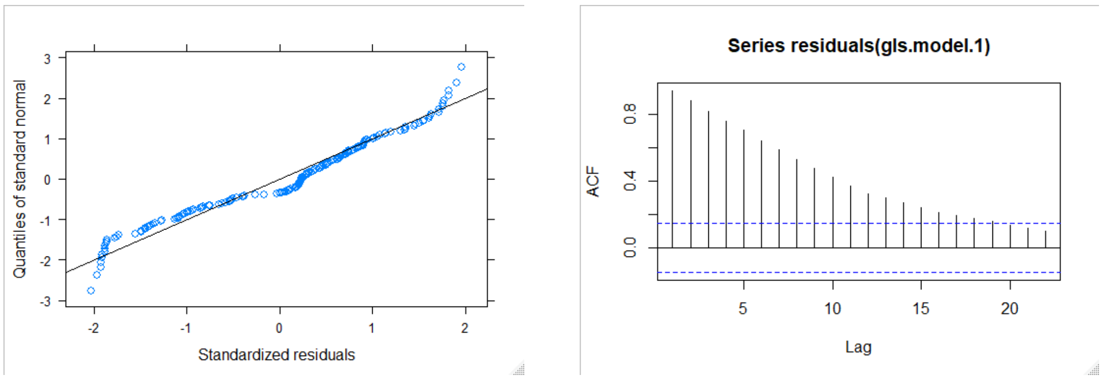

Figure 1:QQ plot and ACF plot of residuals of the segmented regression model for digital volume data.

Similar steps are followed for digital value data by fitting an appropriate segmented regression model. The residuals for this model also show autocorrelation and are significantly different from white noise.

```
gls.model.2 <- gls(Percentage ~ T + X + Ti,
                   data = digital_value_perc_lm_df,
                   correlation = corARMA(form = ~ T, p = 1, q = 1),
                   method = "ML")
```

Figure 2 shows the QQ plot and ACF plot of the model residuals for the digital value data.

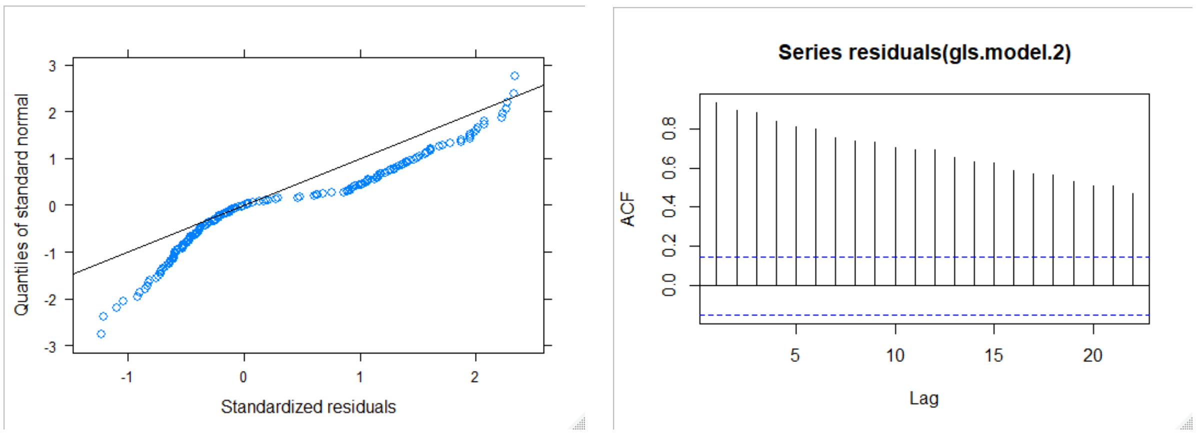

Figure 2:QQ plot and ACF plot of residuals of the segmented regression model for digital value data.

5.6 Time series analysis

Here, the data is converted to time series by using the ts() method as shown below:

```
digi_vol_perc_ts <- ts(digital_volume_perc_df, start = c(2005, 4), frequency = 12)
```

The start date as mentioned as April 2005 because from the plots from EDA, it was seen that for the first 12 months, the digital volume percent was almost 100%. This might be because of no records present for other types of payments.
Next, the stationarity of time series data is checked using ADF and KPSS tests as shown below:

```
adf.test(digi_vol_perc_ts)
kpss.test(digi_vol_perc_ts)
```

The ADF test returns a p-value of 0.8024 and the KPSS test return a p-value of 0.01, both indicating that the series is not stationary.
The ndiffs() function is used to determine the number of differences required to make the series stationary.

```
ndiffs(digi_vol_perc_ts)
digi_vol_perc_ts_diff1 <-  diff(digi_vol_perc_ts)
```

After taking the first difference, the series becomes stationary as indicated by ADF and KPSS tests. This shows that the differencing parameter d = 1 should be used in the Arima models. For automatic detection of the ARIMA order, auto.arima() function is used. In cases where the automatic order detection does no detect d = 1, it is specified explicitly.
The series is decomposed to view the trend and seasonal components.

```
dec_tot_vol_per <- decompose(digi_vol_perc_ts)
```

The data is seasonally adjusted before further processing.

```
deseasonal_tot_vol_per <- seasadj(dec_tot_vol_per)
```

The data is split into train and test series as shown in the code below:

```
digi_vol_perc_ts_train <- window(deseasonal_tot_vol_per, end = c(2018, 10),
frequency = 12)
digi_vol_perc_ts_test <- window(deseasonal_tot_vol_per, start = c(2018, 11),
frequency = 12)
```

Different models are estimated as described below.

5.6.1 Dynamic Regression Models

In this step, an ARIMA model with an external regressor is estimated for the digital volume and digital value percentage data. The external regressors are determined in two ways: 1) detect break points in the data and if a breakpoint is found around the demonetisation time then use it to create a variable with value 0 before and 1 after the breakpoint index; 2) detect outliers in the data and use the level shift and additive outliers as external regressors.

5.6.1.1 Using breakpoints to create external regressors

To determine if there was change in the trend or mean level of the series caused by demonetisation, the series is analysed using changepoint analysis using the strucchange package. The breakpoints() function is used to determine breaks in the series as shown below.

```
break_point <- breakpoints(digi_vol_perc_ts_train ~ 1)
summary(break_point)
plot(digi_vol_perc_ts_train)
lines(fitted(break_point, breaks = 5), col = 4)
lines(confint(break_point, breaks = 5))
```

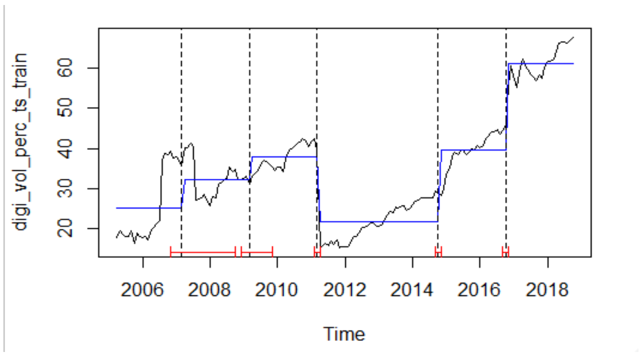

Figure 3:Break points in the digital volume data.

Figure 3 shows that there is a breakpoint at the end of 2016. Use the index of this breakpoint to create a variable with 0 before and 1 after it, as follows:

```
ind_post = break_point$breakpoints[length(break_point$breakpoints)]
xreg <-  c(rep(0,ind_post), rep(1, length(digi_vol_perc_ts_train)-ind_post))
```

Use this variable as an external regressor and fit an ARIMA model to the data. Use the auto.arima() function from the forecast package and set the stepwise and approximation parameters as False to prevent the function from skipping orders or approximating the information criteria while selecting the best model.

```
model1.t.vol <- auto.arima(digi_vol_perc_ts_train,
                           xreg = xreg,
                           method = "ML",
                           stepwise = F,
                           approximation = F,
                           trace = T)
```

The fitted model is subject to diagnostic checks by using the checkresiduals() function.

```
cbind("Regression Errors" = residuals(model1.t.vol, type="regression"),
      "ARIMA errors" = residuals(model1.t.vol, type="innovation")) %>%
autoplot(facets=TRUE)
checkresiduals(model1.t.vol)
Ljung-Box test
data:  Residuals from Regression with ARIMA(1,0,0)(0,0,1)[12] errors
Q* = 5.4263, df = 20, p-value = 0.9995
Model df: 4.   Total lags used: 24
```

The graphs in Figure 4 show the regression errors and errors from ARIMA model.

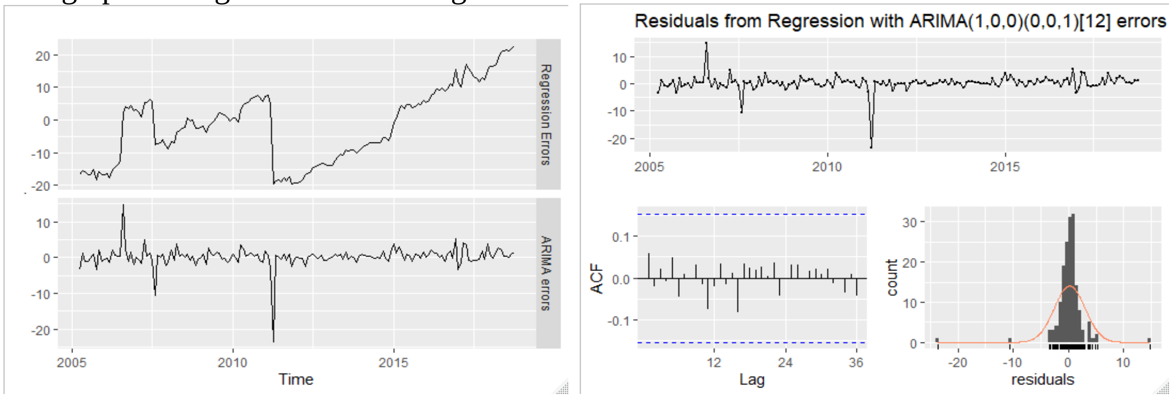

Figure 4: Residuals from Regression with ARIMA(1,0,0)(0,0,1)[12] errors model.

The ARIMA errors follow a white noise pattern. The ACF plot of residuals shows that there is no significant autocorrelation and the p-value of the Ljung-Box test indicates that the residuals are not significantly different from white noise.

Forecast the values for the next 12 months by using the forecast() function with the xreg parameter as a vector of 1s, since the level shift will continue, and the h parameter set to 12.

```
fcast <- forecast(model1.t.vol, xreg=rep(1,12), h=12)
```

Calculate the accuracy of forecasts by using the accuracy() function from the forecast package.

```
accuracy(fcast, digi_vol_perc_ts)
```

Modify the order of ARIMA and fit different models. Compare the AIC, AICc and BIC values and select the model with the lowest values. Compare the forecast accuracy for the test sets using different model fits as shown for the above model and determine the best model based on least value of error measures like RMSE, MAE, MAPE and MASE.

Perform the same steps for digital value data to determine the best model.

```
(break_point <- breakpoints(digi_value_perc_ts_train ~ 1))
# Create a level shift variable with value = 0 pre-intervention and 1 post-
intervention
ind_post = break_point$breakpoints[length(break_point$breakpoints)]
xreg <-  c(rep(0,ind_post), rep(1, length(digi_value_perc_ts_train)-ind_post))

# Fit an ARIMA model with xreg to account for intervention
model1.t.val <- auto.arima(digi_value_perc_ts_train,
                           xreg = xreg,
                           method = "ML",
                           stepwise = F,
                           approximation = F,
                           trace = T)
#Diagnostic checks
checkresiduals(model1.t.val) # residuals are not significantly different from white
noise
#Forecast
fcast <- forecast(model1.t.val,
                  xreg=rep(1,12), h=12)
model1.t.val.acc <- accuracy(fcast, digi_value_perc_ts_test)
```

5.6.1.2 Outliers as external regressors

To detect outliers in the time series data, the tso() function from the tsoutliers package is used.

```
outliers_t.vol <- tso(digi_vol_perc_ts_train, types = c("TC", "AO", "LS", "IO",
"SLS"))
plot(outliers_t.vol)
```

Figure 5 shows the outliers in digital volume data and shows the presence of level shift outliers at the end of 2016.

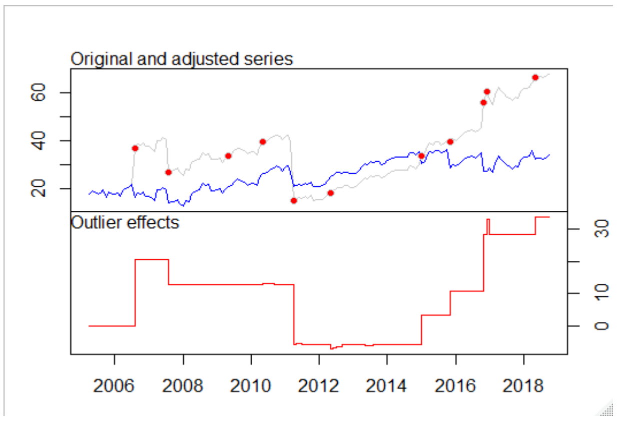

Figure 5: Outliers in the digital volume data.

Find the effect of the Level Shift and Additive Outliers at each data point in the series.

```
# Level shift outlier at the indices
ls_index <- outliers_t.vol$outliers$ind[which(outliers_t.vol$outliers$type ==
'LS')]
ls_outliers <- outliers("LS", ls_index)
#Find effect at each point in the time series
n <- length(digi_vol_perc_ts_train)
ls_effect <- outliers.effects(ls_outliers, n)

# Additive outlier at the indices
ao_index <- outliers_t.vol$outliers$ind[which(outliers_t.vol$outliers$type ==
'AO')]
ao_outliers <- outliers("AO", ao_index)
#Find effect
ao_effect <- outliers.effects(ao_outliers, n)
```

Create a matrix of these outliers and use as external regressors in ARIMA.

```
xreg.outliers <- cbind(ls_effect, ao_effect)
model4.t.vol <- auto.arima(digi_vol_perc_ts_train,
                           stepwise = F,
                           approximation = F,
                           trace = T,
                           xreg = xreg.outliers)
```

The function auto.arima() detects the best model as Regression with ARIMA(0,0,4)(0,0,1)[12] errors.
Check residuals of the model.

```
checkresiduals(model4.t.vol)
Ljung-Box test
data:  Residuals from Regression with ARIMA(0,0,4)(0,0,1)[12] errors
Q* = 136.6, df = 10, p-value < 2.2e-16
Model df: 14.   Total lags used: 24
```

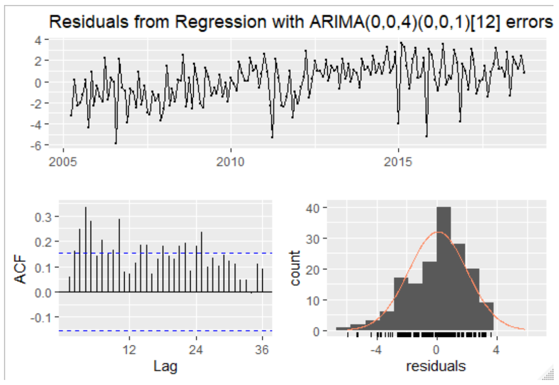

Figure 6: Residuals from Regression with ARIMA(0,0,4)(0,0,1)[12] errors

ACF plot of residuals in Figure 6 shows that autocorrelation is present and Ljung-box test p-value indicates that residuals are not white noise. ndiffs() function indicated that differencing of order 1 is required, therefore pass the parameter d = 1 in auto.arima().

```
model5.t.vol <- auto.arima(digi_vol_perc_ts_train,
                           stepwise = F,
                           approximation = F,
                           trace = T,
                           xreg = xreg.outliers, d = 1)
```

This returns Regression with ARIMA(0,1,1)(1,0,1)[12] errors as the best model. Residual checks indicate that residuals are not significantly different from white noise as shown in Figure 7.

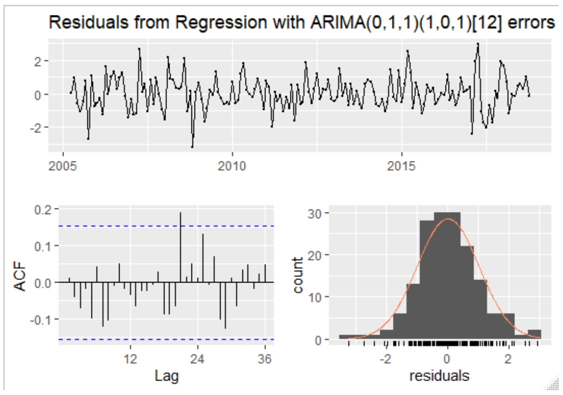

Figure 7: Residuals from Regression with (0,1,1)(1,0,1)[12] errors model

Use the model to forecast for the next 12 months using the forecast() function. The future values of xreg parameter are calculated as shown below. For level shifts, the future values will be 1 and for additive outliers, the future values will be 0.

```
LS17 <- rep(1,12)
LS29 <- rep(1,12)
LS73 <- rep(1, 12)
LS118 <- rep(1, 12)
LS128 <- rep(1, 12)
LS140 <- rep(1, 12)
LS158 <- rep(1,12)
AO141 <- rep(0, 12)

fut.xreg <- cbind(LS17, LS29, LS73, LS118, LS128, LS140, LS158, AO141)
fut.xreg <- as.matrix(fut.xreg)

fcast <- forecast(model5.t.vol,
                  xreg = fut.xreg, h=12)
```

Plot the forecasts and actual values.

```
plot(fcast, main = "Digital transaction Volume forecasts", sub = "Regression with
ARIMA(0,1,1)(1,0,1)[12] errors" ,
     xlab = "Time", ylab = "Percentage")
lines(digi_vol_perc_ts_test, col = "red")
legend(2005, 75, legend = c("Actual", "Predicted"), col = c("red", "blue"), lty = 1
, cex= 0.8)
```

Calculate the accuracy of forecasts.

```
accuracy(fcast, digi_vol_perc_ts_test)
```

Modify the order of ARIMA and fit different models. Compare the AIC, AICc and BIC values and select the model with the lowest values. Compare the forecast accuracy for the test sets using different model fits and determine the best model based on least value of error measures like RMSE, MAE, MAPE and MASE.
Perform the same steps for digital value data to determine the best model.

```
outliers_t.val <- tso(digi_value_perc_ts_train, types = c("TC", "AO", "LS", "IO", 
"SLS"))
n <- length(digi_value_perc_ts_train)

# Level shift outlier at the indices
ls_index <- outliers_t.val$outliers$ind[which(outliers_t.val$outliers$type == 
'LS')]
ls_outliers <- outliers("LS", ls_index)
ls_effect <- outliers.effects(ls_outliers, n)

# Additive outlier at the indices
ao_index <- outliers_t.val$outliers$ind[which(outliers_t.val$outliers$type == 
'AO')]
ao_outliers <- outliers("AO", ao_index)
ao_effect <- outliers.effects(ao_outliers, n)

xreg.outliers <- cbind(ls_effect, ao_effect)

# Modelling with ARIMA with outliers as xreg variables
model4.t.val <- auto.arima(digi_value_perc_ts_train,
                           stepwise = F,
                           approximation = F,
                           trace = T,
                           xreg = xreg.outliers)
# Diagnostic checks
checkresiduals(model4.t.val)

#Forecast
LS92 <- rep(1,12)
AO73 <- rep(0, 12)
fut.xreg <- cbind(LS92, AO73)
fut.xreg <- as.matrix(fut.xreg)

fcast <- forecast(model4.t.val,
                  xreg = fut.xreg, h=12)
```

5.6.2 ARIMA models with transfer functions

To fit at an ARIMA model with transfer function, the ARIMA order of the pre-intervention series is detected by using auto.arima() function with parameters stepwise and approximation set to False.

```
auto.arima(window(digi_vol_perc_ts_train, end = c(2016,10)),
           stepwise = F,
           approximation = F,
           trace = T)
```

This order is then used on the entire training series along with the transfer function parameters passed to the arimax() function. In the code shown below, the xtransf parameter is used to specify an external covariate which is a pulse indicator at index 140. It is known that demonetisation occurred at time point 140 in the series. Therefore, the covariate is created such that it has the value 1 at index 140 and 0 otherwise, indicating a temporary change. The transfer parameter contains a list of (p,q) which determines the AR and MA order of the covariate. Here, this value is set as (0,0) to specify that no lagged effects of covariates are taken into consideration. The estimation method used is Maximum Likelihood estimation.

```
# Pulse function
model7.t.vol <- arimax(digi_vol_perc_ts_train, order = c(0,1,0),
                       seasonal = list(order=c(0,0,1), period=12),
                       xtransf = data.frame(Nov16 = 1 *
(seq_along(digi_vol_perc_ts_train) == 140)),
                       transfer = list(c(0,0)),
                       method = "ML")
```

Perform diagnostic tests on the model residuals.

```
checkresiduals(model7.t.vol)
Ljung-Box test
data:  Residuals from ARIMA(0,1,0)(0,0,1)[12]
Q* = 6.6342, df = 22, p-value = 0.9993
Model df: 2.   Total lags used: 24
```

The residuals represent white noise.

Various transfer functions are estimated by changing the covariates and transfer lists. The step function is modelled by setting the covariate as 0 before and 1 after demonetisation.

```
#Step function
model8.t.vol <- arimax(digi_vol_perc_ts_train, order = c(0,1,0),
                       seasonal = list(order=c(0,0,1), period=12),
                       xtransf = data.frame(Nov16 = 1 *
(seq_along(digi_vol_perc_ts_train) >= 140)),
                       transfer = list(c(0,0)),
                       method = "ML")
```

A pulse function with AR(1) effect is modelled as shown below. This is used to determine if the lagged values of the covariate are significant in affecting the actual series.

```
# Pulse function with AR(1) --> effect gradually dies out
model9.t.vol <- arimax(digi_vol_perc_ts_train, order = c(0,1,0),
                       seasonal = list(order=c(0,0,1), period=12),
                       xtransf = data.frame(Nov16 = 1 *
(seq_along(digi_vol_perc_ts_train) == 140)),
                       transfer = list(c(1,0)),
                       method = "ML")
```

A step function with AR(1) is modelled as shown below. The lagged values of the covariate affect the series, indicating that there is a gradual change in the mean level of the series.

```
# Step Function with AR(1) -- intervention affects the mean of the series gradually
model10.t.vol <- arimax(digi_vol_perc_ts_train, order = c(0,1,0),
                        seasonal = list(order=c(0,0,1), period=12),
                        xtransf = data.frame(Nov16 = 1 *
(seq_along(digi_vol_perc_ts_train) >= 140)),
                        transfer = list(c(1,0)),
                        method = "ML")
```

An instantaneous pulse followed by a step AR(1) transfer function is modelled as shown below.

```
# Instantaneous pulse + step function with AR(1)
model11.t.vol <- arimax(digi_vol_perc_ts_train, order = c(0,1,0),
                        seasonal = list(order=c(0,0,1), period=12),
                        xtransf = data.frame(Nov16 = 1 *
(seq_along(digi_vol_perc_ts_train) == 140),
                                             Nov16 = 1 *
(seq_along(digi_vol_perc_ts_train) >= 140)),
                        transfer = list(c(0,0),c(1,0)),
                        method = "ML")
```

The residuals of all these models are tested using the checkresiduals() function and the best model is selected based on the lowest value of AIC. The best model is found as the step function with AR(1). Coefficients of the model are shown below:

```
Coefficients:
         sma1  Nov16-AR1  Nov16-MA0
      -0.1909     0.3095    10.1734
s.e.   0.0815     0.1948     2.6717
```

To forecast the values for the next 12 months, the future values of the covariates are estimated using the coefficients of the covariates from the model fit. The filter() function is used to calculate the effect.

```
eff <- 1*(seq_along(1:(length(digi_vol_perc_ts_train)+12))>=140)
tf <- filter(eff, filter = 0.309462, method = "recursive", sides = 1) * (10.173352)
```

Since the TSA package does not provide any function to forecast, the Arima() function from the forecast package is used to include the effect of covariates by passing to the xreg parameter. The estimates of this model are used in the forecast() method.

```
fcast <- Arima(digi_vol_perc_ts_train, c(0,1,0),
               seasonal = list(order=c(0,0,1), period=12),
               xreg = tf[1:(length(tf)-12)])
fc <- forecast::forecast(fcast, h = 12, xreg=tf[(length(tf)-11):length(tf)])
```

The forecasted and actual values are plot using the below code.

```
plot(fc, main = "Digital transaction Volume forecasts", sub =
"ARIMA(0,1,0)(0,0,1)[12] with transfer function",
     xlab = "Year", ylab = "Percentage")
lines(digi_vol_perc_ts_test, col = "red")
legend(2005, 85, legend = c("Actual", "Predicted"), col = c("red", "blue"), lty = 1
, cex= 0.8)
```

The impact of demonetisation which is estimated by using the filter() function is visualised using the below code.

```
demonetisation_effect_ts <- ts(tf, frequency = 12, start = c(2005, 04), end =
c(2019, 10))

plot(demonetisation_effect_ts, type = "h",
     main = "Demonetisation effect on Digital Transaction volume",
     xlab = "Year", ylab = "Percent")
```

The magnitude of the impact is found from the model estimates as below.

```
window((demonetisation_effect_ts), start = c(2016, 11))
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
Sep      Oct
2016
2017 14.29589 14.59739 14.69069 14.71956 14.72850 14.73126 14.73212 14.73238
14.73246 14.73249
2018 14.73250 14.73250 14.73250 14.73250 14.73250 14.73250 14.73250 14.73250
14.73250 14.73250
```

```
2019 14.73250 14.73250 14.73250 14.73250 14.73250 14.73250 14.73250 14.73250
14.73250 14.73250
          Nov      Dec
2016 10.17335 13.32162
2017 14.73250 14.73250
2018 14.73250 14.73250
2019
```

This shows that the instantaneous impact in November 2016 was an increase of 10.17%. This grew in the next few months till October 2017 and remained constant thereafter at around 14.73%.

The transfer function modelling for the digital value percentage series is also performed in a similar manner by determining the ARIMA order from the pre-intervention data and estimating the pulse and step functions with and without the effect of lagged values. The best model for this data was also determined based on lowest AIC value. The best model for digital value percentage data is found to be the step function, as shown below.

```
model8.t.val <- arimax(digi_value_perc_ts_train, order = c(3,1,0),
                       seasonal = list(order=c(0,0,1), period=12),
                       xtransf = data.frame(Nov16 = 1 *
(seq_along(digi_value_perc_ts_train) >= 140)),
                       transfer = list(c(0,0)),
                       method = "ML")
```

For forecasting, the model estimates of the covariates are used as shown below.

```
eff <- 1*(seq_along(1:(length(digi_value_perc_ts_train)+12))>=140)
tf <- (eff * (-1.850245))
fcast <- Arima(digi_value_perc_ts_train, order = c(3,1,0),
               seasonal = list(order=c(0,0,1), period=12),
               xreg = tf[1:(length(tf)-12)])

fc <- forecast::forecast(fcast, h = 12, xreg=tf[(length(tf)-11):length(tf)])
```

The forecasts and actual values are plotted using the below code.

```
plot(fc, main = "Digital transaction Value forecasts", sub = "ARIMA(3, 1,
0)(0,0,1)[12] with transfer function" ,
     xlab = "Year", ylab = "Percentage")
lines(digi_value_perc_ts_test, col = "red")
legend(2008, 45, legend = c("Actual", "Predicted"), col = c("red", "blue"), lty = 1
, cex= 0.8)
```

The impact of demonetisation is plotted using the below code.

```
demonetisation_effect_ts <- ts(tf, frequency = 12, start = c(2005, 04), end =
c(2019, 10))

plot(demonetisation_effect_ts, type = "h",
     main = "Demonetisation effect on Digital Transaction Value",
     xlab = "Year", ylab = "Percentage")
```

The magnitude of impact is found as shown below.

```
window(demonetisation_effect_ts, start = c(2016, 11))
Jan       Feb       Mar       Apr       May       Jun       Jul       Aug       Sep
2016
2017 -1.850245 -1.850245 -1.850245 -1.850245 -1.850245 -1.850245 -1.850245 -
1.850245 -1.850245
```

```
2018 -1.850245 -1.850245 -1.850245 -1.850245 -1.850245 -1.850245 -1.850245 -
1.850245 -1.850245
2019 -1.850245 -1.850245 -1.850245 -1.850245 -1.850245 -1.850245 -1.850245 -
1.850245 -1.850245
           Oct       Nov       Dec
2016           -1.850245 -1.850245
2017 -1.850245 -1.850245 -1.850245
2018 -1.850245 -1.850245 -1.850245
2019 -1.850245
```

Thus, demonetisation caused a permanent negative change of around 1.85% on the levels of digital transaction value.