Liver Cancer Forecast

HarvardX Data Science, Machine Learning

C.T. Dinh

March 04, 2023

Introduction

Although cancer incidence and mortality overall are declining in all population groups in the United States, certain groups continue to be at increased risk of developing or dying from particular cancers including breast, prostate, kidney, liver, and lung.

These disparities are frequently seen in people from low-socioeconomic groups, certain racial/ethnic populations, and those who live in geographically isolated areas.

higher rates of liver cancer among Asian and Pacific Islanders than other racial/ethnic groups as stated by Cancer Health Disparities Research at NCI.

Cancer liver has been listed in the Top 20 Disease Sites for Newly Registered in the last 10 years.

For the above mentioned reasons, Liver Cancer Forecast is selected for this project.

The question is not whether cancer mortality rate can be reduced by 50% in 25 years, but rather at what diminishing rate per given year would be optimal in order to meet the cancer reduction goal set by President Biden.

The objectives of this project is to predict liver cancer trend based on historical data by taking advantage of the auto and custom selection algorithm of ARIMA, Simple Exponential Smoothing (SES), and Neural Network time series forecasts to manipulate time series data and get it ready for modeling and forecasting

ARIMA is an acronym for Auto Regressive (AR) Integrated (I) Moving Average (MA):
Brief explanation of the components of ARIMA

Liver Cancer Data

Data on the liver cancer incidence data was obtained from (OCC, 2023) The data contains 790 rows of observations from the annual fiscal year 2009 to 2012. The data will be sorted in chronological order, partitioned according to time into two datasets training data and validation (final_holdout_test) datasets. The modeling approaches will be developed and evaluated using the train and Finally, the model with the best accuracy will be tested using the validation set (final_holdout_test).

Several ARIMA models with different autocorrelation terms will be formulated and chosen one which provided for an accurate fit of the data based on the Akaike information criteria (AIC). A lower AIC would indicate a better model fit. Based on the final selected model, the annual number of cases expected to be registered in the U.S. from 2022 to 2027 will be forecast. The 95% confidence intervals (CIs) will be automatically calculated from the mean square errors of the model.

In summary, this project contains 790 liver cancer cases registered from 2009 to 2021. Model generation will be based on the data from 2009 to 2015 (training dataset) and model validation is based on the dataset 2016 to 2022 (validation dataset). Thereafter,the forecast annual values will be from 2023 to 2027.

Required steps include:

1. Load and Perform exploratory data analysis (EDA)
    - format dataset ISO date, sort, and plot the data and examine its patterns and irregularities
    - clean any outliers using tsclean(), if necessary impute any missing values
    - An article on [Data Cleaning in R Made Simple] (https://towardsdatascience.com/data-cleaning-in-r-made-simple-1b77303b0b17).
2. Decompose the data to see trends and patterns including seasonality in the data.
    - Use decompose() and
    - if there are seasonal signal in the data use stl(), a Season Trend Decomposition using Loess. Note that stl() only has additive seasonal signal and not multiplicative. For more details on multiplicative vs additive time series decompostion.
3. Check whether the observed data is stationary
    - Use adf.test, tsdisplay(), and lag.plot()
4. Partition the data into train & validation according to time
    - Plot the two data series
5. Create auto and custom best fitted ARIMA models for forecasting
    - Examine the results of various model fitting using tools such as summary(), tsdispaly(), ACF(), PACF() for any lags/gaps
    - Visually examine the fitted model against the observed data via plot.
    - Evaluate each model for errors or residuals and accuracy using tools such as checkresiduals(), tsdisplay(residuals()), or ets()
    - repeat the whole process
6. Forecast the best fitted model against the validation data series (hold-out-set).
7. Conclusion
    - Lessons Learned
    - Future or additional work

Prepare Required Packages

```
# Required packages for analysis
pkg <- c(
  "caret", "tidyverse", "knitr", "styler", "broom", "data.table", "dplyr", "ggplot2",
  "gghighlight", "kableExtra", "pagedown", "readr", "stringr", "scales", "gridExtra",
  "tseries", "lubridate", "formattable", "smooth", "ggfortify", "grid"
)

# Check if packages are not installed and assign the names of the packages not installed to th
e variable new.pkg
new.pkg <- pkg[!(pkg %in% installed.packages())]

# If there are any packages in the list that aren't installed, install them
if (length(new.pkg)) {
  install.packages(new.pkg, repos = "http://cran.rstudio.com")
}

# Load the libraries
library(caret) # createTimeSlices
library(knitr) # for knit, kable, lightweight API's designed to give users full control of the
 output without heavy coding work.
library(tidyverse)
library(styler) # cleanup messy code with the styler addin
library(broom) # broom and kableExtra packages produce beautiful tables
library(data.table)
library(dplyr) # for data manipulation (eg inner_join, merge)
library(gghighlight)
library(ggplot2)
library(ggthemes)
library(kableExtra) # for beautifying HTML output
library(pagedown) # for converting from html to pdf
library(readr) # for read_csv
library(stringr)
library(scales) # for converting y/x axis label with scientific notation or comma separator
library(gridExtra) # for providing useful extensions to the grid system, i.e. add a table grid
 inside a ggplot
library(ggfortify) # for autoplot(), extends ggplto2 for plotting
library(forecast) # For ARIMA() function
library(tseries) # for time series partitioning and objects
library(lubridate) # for fast and user friendly parsing of date-time data
library(formattable) # for formatting decimal places
```

1. Load the Data

Load the data and perform exploratory data analysis (EDA) process includes format, sort,and examine the data structurally and visually. Instructions on how to get raw data from github, see this link.

```
# set working dir
setwd(dir = "C:/Chi/HarvardXCYO/")

# All defaults
img_path <- "C:/Chi/HarvardXCYO/images/"

# download the data (liver cases) file from github:
urlfile <- "https://raw.githubusercontent.com/STEMenerChi/DataScience/main/HarvardXCYOProject/
regByLiver.csv"
# set stringsAsFactors = FALSE so that the string won't get converted into factor
dataL <- read.csv(urlfile, stringsAsFactors = FALSE)

# download data (liver cases by fy)
urlfile2  <-  "https://raw.githubusercontent.com/STEMenerChi/DataScience/main/HarvardXCYOProject
/regByFY.csv"
dataByFY <- read.csv(urlfile2, stringsAsFactors = FALSE)


# Convert FY into ISO date format
dataL$as.date <- as.Date(as.character(dataL$fy), format = "%Y")

# It is important to sort the data in a chronological order before convert it into a time seri
es (TS) object
# the date does not go into the TS object, only 3 parameters: begin date, end date and frequen
cy.
dataL <- dataL[order(dataL$as.date), ]
```

Examine data structure. The data contains 790 rows of observations from the annual fiscal year 2009 to 2021 and 5 variables, as described below:

```
    1. fy - fiscal year start from 2009 to 2021
    2. id - data source identification number
    3. cancersite - cancer disease sites, for this project it's "liver" cancer
    4. regpatient - number of registered patients (dependent variable)
    5. as.date - converted fy into as.date for time series
```

fy and regpatient will be the focal points in this project.

```
str(dataL)
```

```
## 'data.frame':     790 obs. of  5 variables:
##  $ fy        : int  2009 2009 2009 2009 2009 2009 2009 2009 2009 2009 ...
##  $ id        : int  1 2 3 4 6 7 8 9 11 13 ...
##  $ cancersite: chr  "Liver" "Liver" "Liver" "Liver" ...
##  $ regpatient: int  200 29 110 137 39 70 81 130 125 69 ...
##  $ as.date   : Date, format: "2009-03-04" "2009-03-04" ...
```

There are 13 fiscal years (FY):

```
unique(dataL$fy)
```

```
##  [1] 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021
```

List first 7 and last 7 rows of data:

```
dataL %>%
  {
    rbind(head(., 7), tail(., 7))
  } %>%
  kbl(caption = "First and Last 7 Rows of Data") %>%
  kable_classic_2(full_width = F, c("striped", "hover"))
```

First and Last 7 Rows of Data

	fy	id	cancersite	regpatient	as.date
1	2009	1	Liver	200	2009-03-04
2	2009	2	Liver	29	2009-03-04
3	2009	3	Liver	110	2009-03-04
4	2009	4	Liver	137	2009-03-04
5	2009	6	Liver	39	2009-03-04
6	2009	7	Liver	70	2009-03-04
7	2009	8	Liver	81	2009-03-04
784	2021	65	Liver	107	2021-03-04
785	2021	66	Liver	176	2021-03-04
786	2021	68	Liver	230	2021-03-04
787	2021	72	Liver	101	2021-03-04
788	2021	79	Liver	206	2021-03-04
789	2021	85	Liver	49	2021-03-04
790	2021	87	Liver	95	2021-03-04

Examine liver cancer cases per FY:

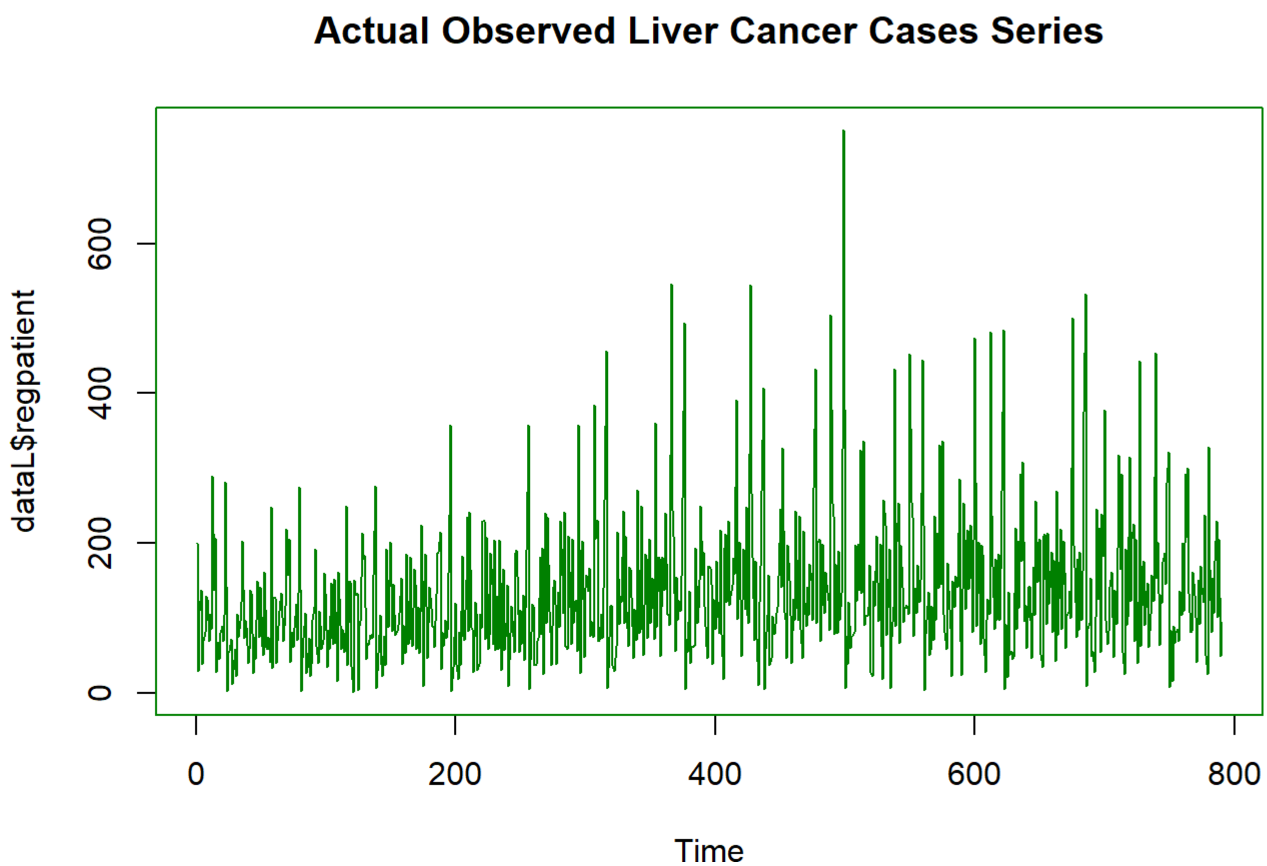
```
dataByFY %>%
  kbl(caption = "Cases per FY") %>%
  kable_classic_2(full_width = F, c("striped", "hover"))
```

Cases per FY

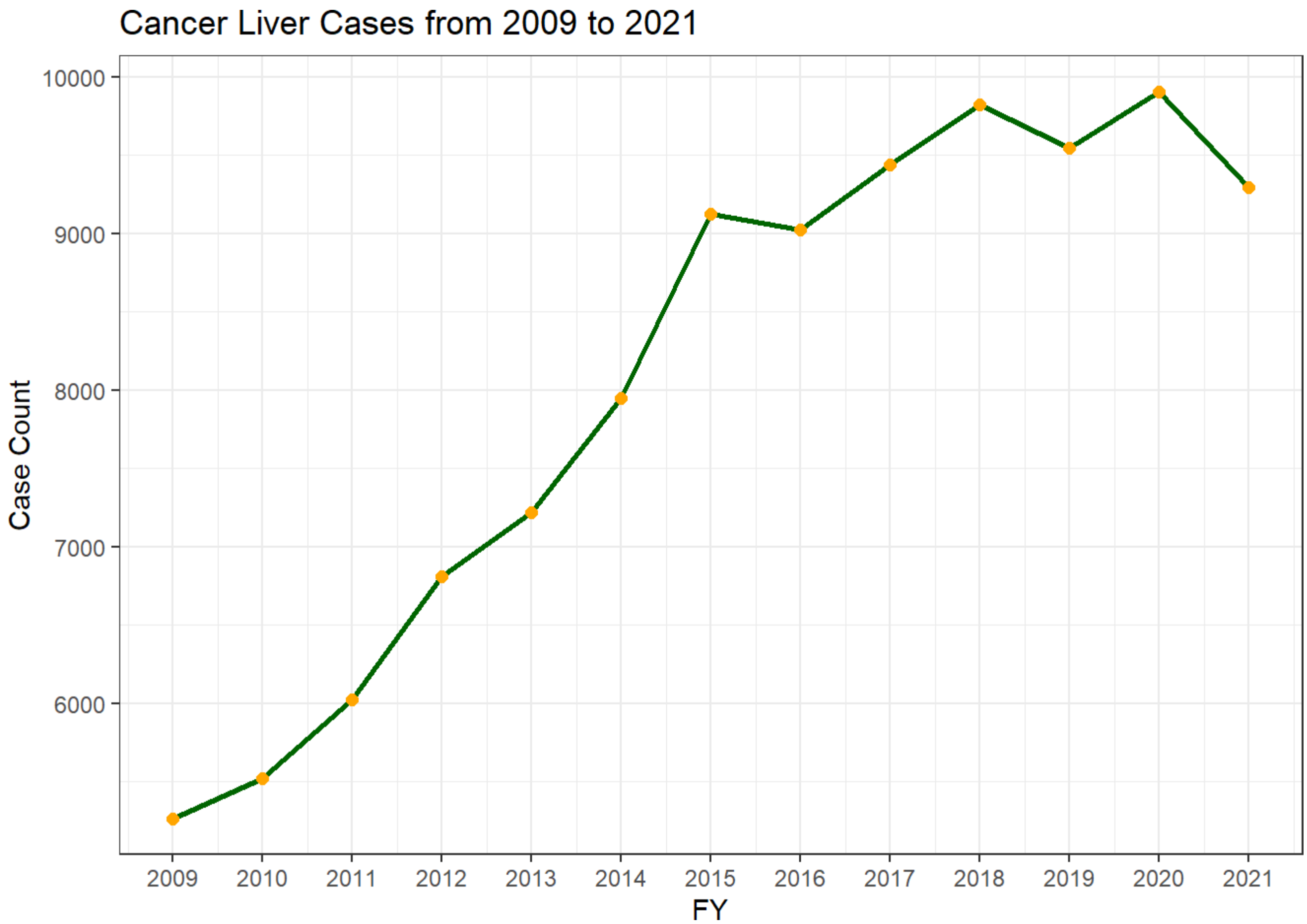
fy	Liver_ase
2009	5267
2010	5524
2011	6026
2012	6809
2013	7223
2014	7948
2015	9124
2016	9023
2017	9443
2018	9821
2019	9549
2020	9908
2021	9297

Visually examine liver cancer case time series

```
par(col = "#008000")
plot.ts(dataL$regpatient, main = "Actual Observed Liver Cancer Cases Series")
```




```
# cases count by year
plt <- dataByFY %>%
  ggplot(aes(x = fy, y = regpatient)) +
  geom_line(color = "darkgreen", lwd = 1) +
  geom_point(color = "orange", lwd = 2) +
  theme_bw() +
  ggtitle("Cancer Liver Cases from 2009 to 2021") +
  xlab("FY") +
  ylab("Case Count") +
  scale_x_continuous(breaks = 2009:2021)
```



The number of liver cancer cases progressively increased over the years, except there are dips in 2016, 2019, and 2021. The table below shows the overview of the number of cases, average count, and percentage of case changes from year to year:

Liver Cancer Cases Overview

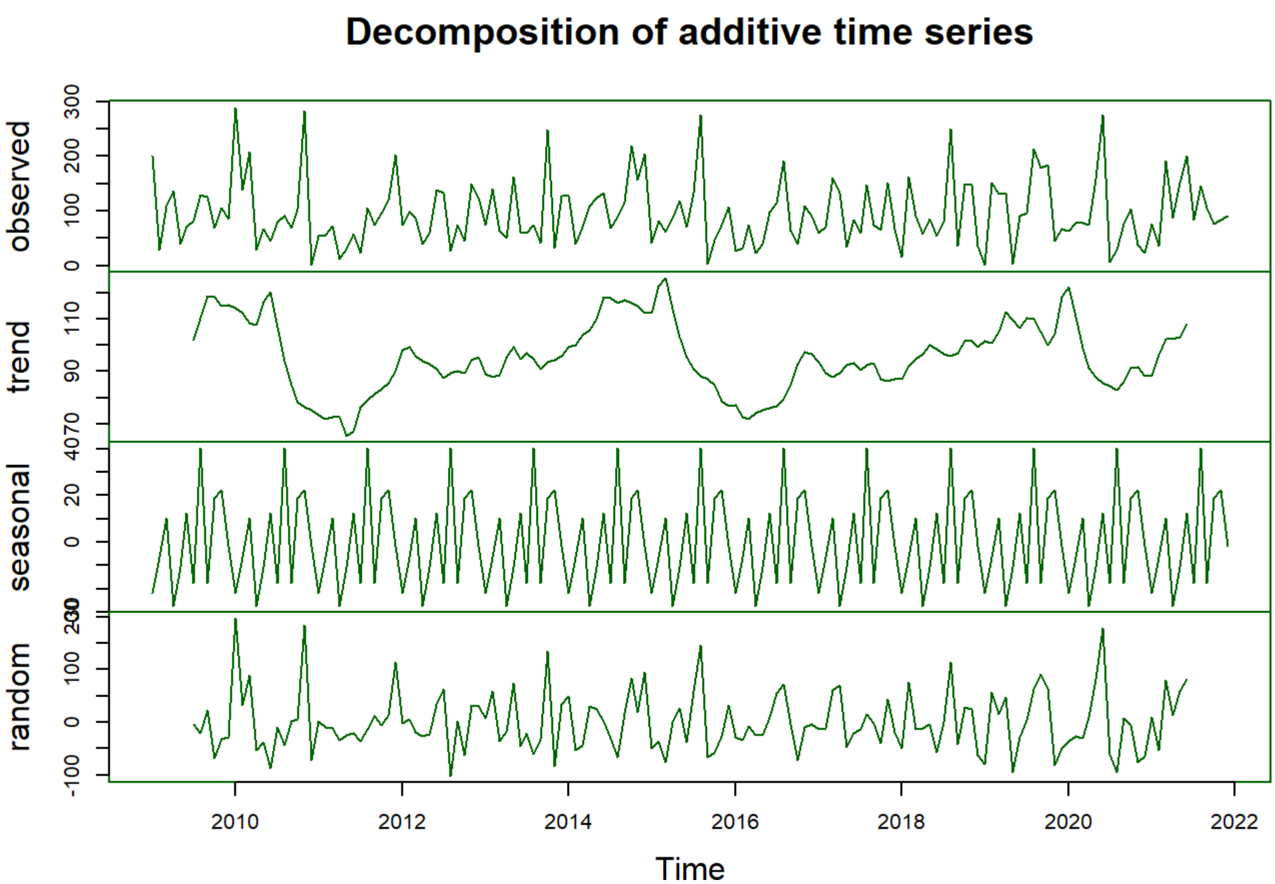
fy	case_count	avg_count	percent_change
2009	5267	92.4	NA
2010	5524	95.2	4.9
2011	6026	103.9	9.1
2012	6809	113.5	13.0
2013	7223	118.4	6.1
2014	7948	134.7	10.0
2015	9124	147.2	14.8
2016	9023	147.9	-1.1
2017	9443	154.8	4.7
2018	9821	158.4	4.0
2019	9549	151.6	-2.8
2020	9908	154.8	3.8
2021	9297	145.3	-6.2

2. Decompose the Data

Using decompose () function from base R to visually examine trends and patterns including seasonality in the data in four individual O, T, S, and R components:

- The first graph is the **O**bserved data,
- the second is the **T**rend which is the moving average (MA),
- the third is **S**easonal signals without the irregular fluctuations involved, and
- the last graph is the **R**andom signals those are general fluctuations in the data that cannot be accounted for.

```
# convert data into time series object
dataL.ts <- (ts(dataL$regpatient, start = c(2009, 1), end = c(2021, 12), freq = 12))
# decompose data
par(col = "darkgreen")
decomp_add <- decompose(dataL.ts, type = "additive")
plot(decomp_add)
```



3. Check Data for Stationary

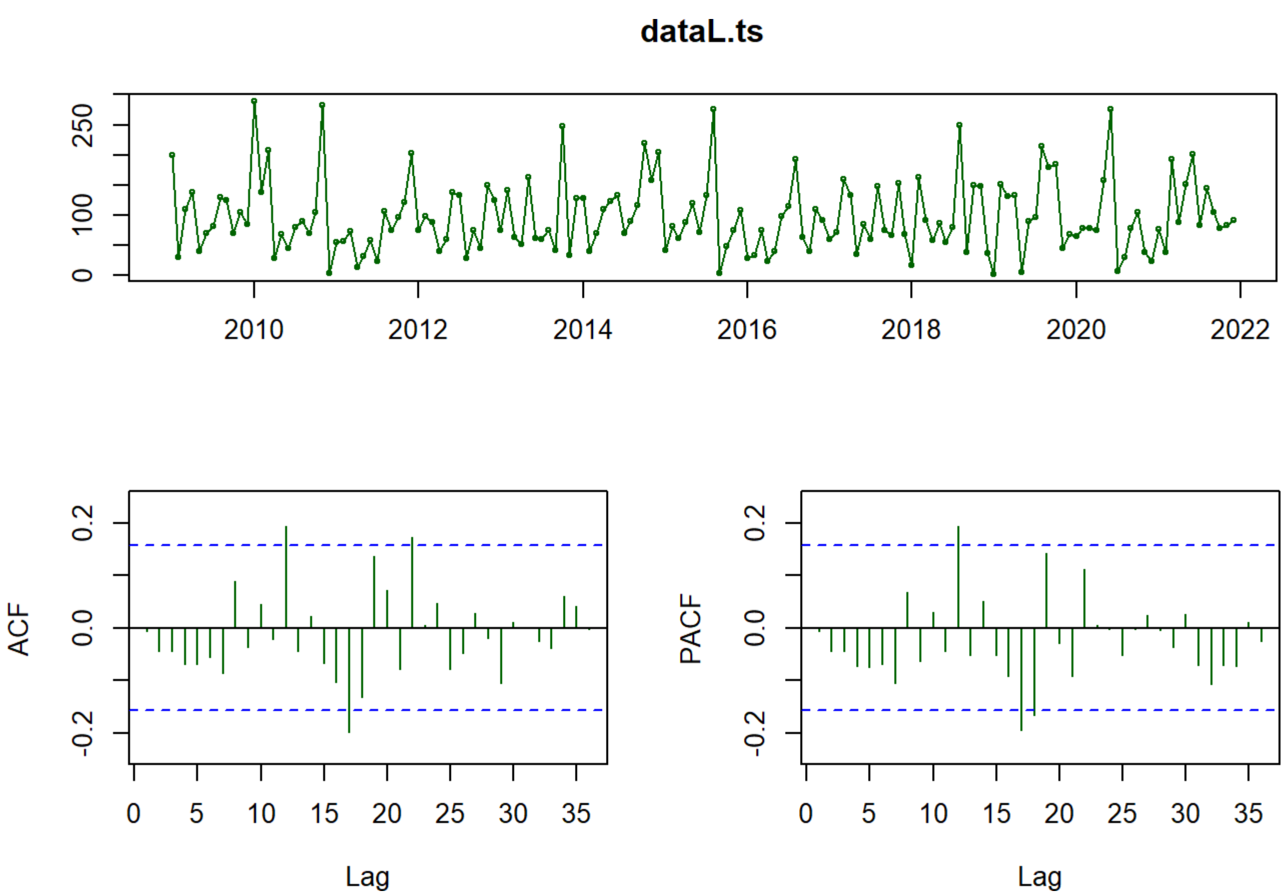
Stationarity is an important concept in the field of time series (TS) analysis with tremendous influence on how the data is perceived and predicted. When forecasting or predicting the future, each point is independent of one another in most TS models. The augmented dickey fuller (ADF) test is a common test in statistics and is used to check whether a given TS is stationary or at rest if it doesn’t have any trend and depicts a constant variance over time and follows autocorrelation structure over a period constantly. The more negative magnitude of the ADF number is, the stronger the rejection of the hypothesis that there is a unit root at some level of confidence.

```
adf.test(dataL.ts)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  dataL.ts
## Dickey-Fuller = -5.9906, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Dickey-Fuller returns negative value confirms that TS is stationary. In addition, the p-value is less than 0.05 is typically considered to be statistically significant, in which case the null hypothesis should be rejected, concluded that this TS is stationary. The data series is ready to be analyzed.

```
forecast::tsdisplay(dataL.ts, col = "darkgreen")
```



The ACF plots the correlation coefficient against the lag, which is measured in terms of a number of periods or units. The blue dashed lines represent an approximate confidence interval (CI) for what is produced by white noise, by default the lines are displaying the 95 CI. Anything displays above the blue line is notably strong; anything displays below is not distinguishable from zero.

If we have strong peeks that means we definitely have autocorrelation structure in our data. From visual assessment, our time plots do not show trends or seasonality which is considered stationary.

Based on the ACF graph, there are lags at time step 12 and 22, these lags will be addressed later in ARIMA models. The partial autocorrelation function (PACF) confirms that there is a lag at time step 12.

4. Partition Time Series Data

Now that it's confirmed that the data is stationary. The time series data will be evenly split according to time into training from 2009-2015 and validation from 2015-2021. The 'start' and 'end' arguments specifies the time of the first and the last observation, respectively. The argument 'frequency' specifies the number of observations per unit of time. In case it's 12 months.

```
# check for min and max date
min_date <- min(dataL$as.date)
max_date <- max(dataL$as.date)

# Build a time series data
dataL.ts <- ts(dataL$regpatient, start = c(2009, 1), end = c(2021, 12), freq = 12)
# dataL.ts

# Evenly Split the data series into train and test sets according to time
# Both train and valid contain 2015 data
trainL.ts <- window(dataL.ts, start = c(2009, 1), end = c(2015, 12), freq = 12)
validL.ts <- window(dataL.ts, start = c(2015, 1), end = c(2021, 12), freq = 12)


trainL.ts
```

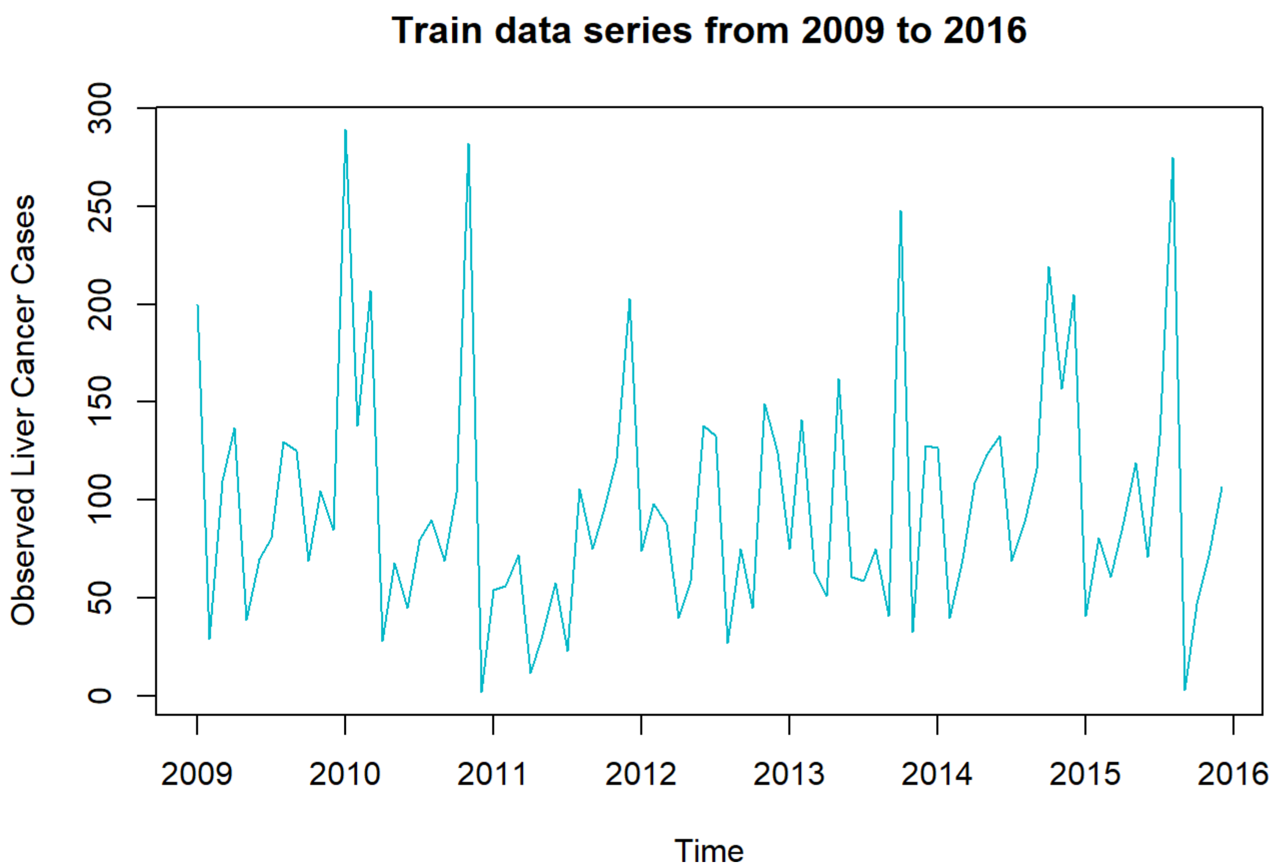
```
##       Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2009 200  29 110 137  39  70  81 130 125  69 105  85
## 2010 289 138 207  28  68  45  80  90  69 104 282   2
## 2011  54  56  72  12  31  58  23 106  75  96 121 203
## 2012  74  98  88  40  59 138 133  27  75  45 149 124
## 2013  75 141  63  51 162  61  59  75  41 248  33 128
## 2014 127  40  70 109 123 133  69  89 116 219 157 205
## 2015  41  81  61  87 119  71 133 275   3  47  74 107
```

```
validL.ts
```

```
##       Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2015  41  81  61  87 119  71 133 275   3  47  74 107
## 2016  27  33  74  23  39  98 115 192  63  40 110  91
## 2017  59  71 160 133  34  84  59 147  74  66 152  67
## 2018  16 162  91  58  86  55  80 250  37 150 148  36
## 2019   1 151 131 133   4  90  96 214 179 184  45  67
## 2020  64  78  78  74 158 276   6  29  77 104  38  23
## 2021  76  37 192  88 151 201  83 145 105  77  83  91
```

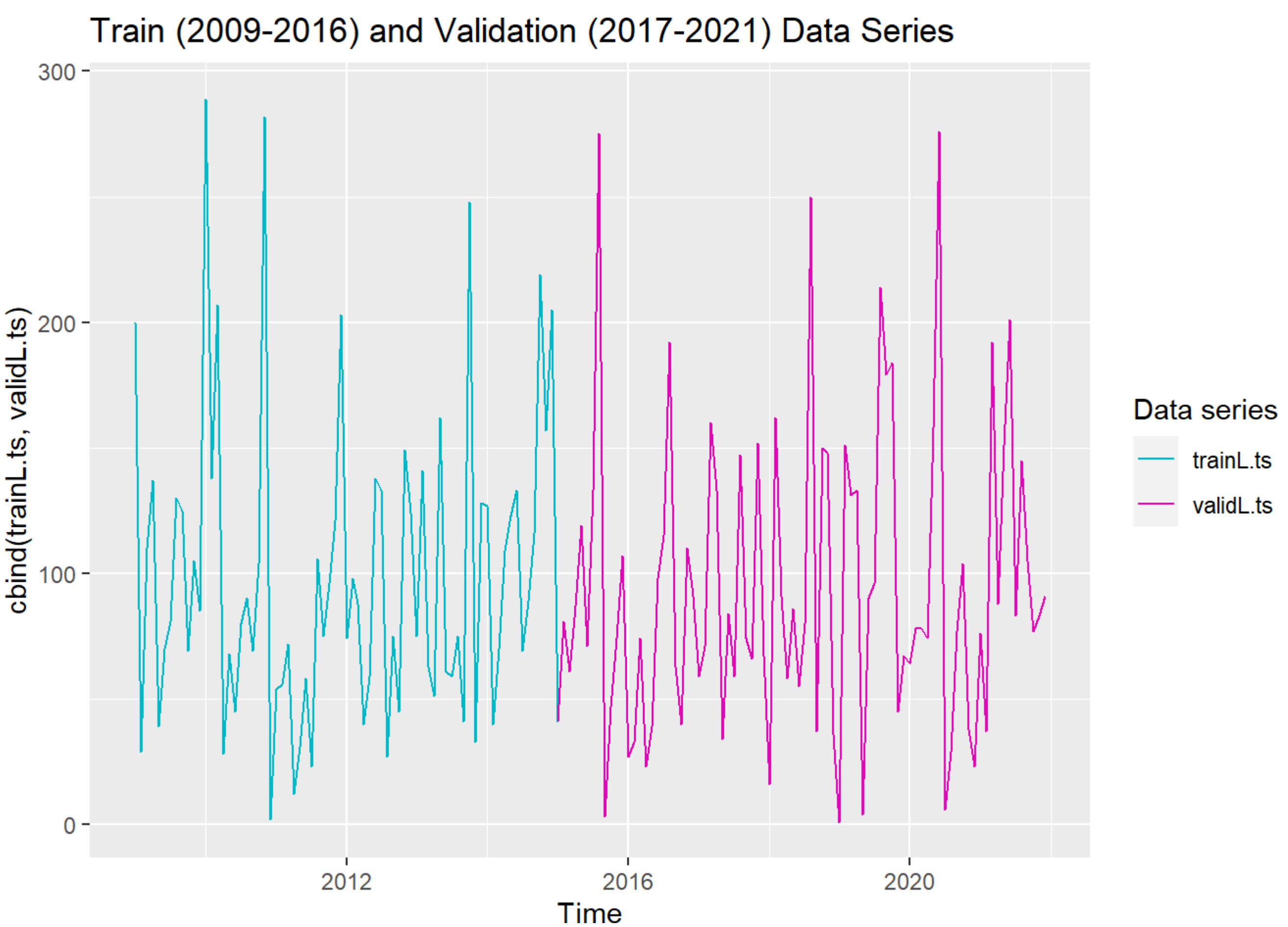
Training data series plot:

```
# Plot the train data series:
plot(trainL.ts, col = "#00B7C7", ylab = "Observed Liver Cancer Cases", main = "Train data seri
es from 2009 to 2016")
```



Both training (from 2009-2015) and validation (2015-2021) data series plot:

```
# Plot both the train and validation data series
autoplot(cbind(trainL.ts, validL.ts)) +
  ggtitle("Train (2009-2016) and Validation (2017-2021) Data Series") +
  guides(colour = guide_legend(title = "Data series")) +
  scale_colour_manual(values = c("#00B7C7", "#dc0ab4"))
```



5. Create & Evaluate Models

Several models including ARIMA (auto and custom) will be fitted and evaluated.

An autoregressive integrated moving average (ARIMA) is a statistical analysis model that predicts future values based on past values. The default auto.arima() shows non-seasonal and seasonal:

For nonseasonal= c(p, d, q) a lowercase p for autoregressive component a lowercase d for differencing component a lowercase q for MA component.

Uppercase P, D, Q are used for seasonal = c(P, D, Q). Max default values for seasonal is c(2,1,2) for **nonseasonal is c(5,2,5)**.

A seasonal pattern occurs when a time series is affected by seasonal factors such as the time of the year or the day of the week. Seasonality is always of a fixed and known frequency. Since there is no seasonal signals or pattern in our data, we will only focus on ARIMA(p,d,q) parameterization in our model selection.

The residuals in ARIMA models tell a story about the performance of the model and should be taken into consideration when evaluating them. The functions such as checkresiduals, ACF and PACF will be used to keep track of the information left behind in the residuals by the model.

Using the **training ts**, iterate through these steps:

a. Fit the model
b. Plot the model
c. Check for coefficients and error measures in the model using summary()
d. Check for p-value of the model using checkresiduals()
e. Forecast the model
f. Plot the forecast model on the observed ts
g. Check for lags, examine ACF and PACF using tsdisplay()
h. select another model

repeat steps a-h.

Initialize the forecast term to 5 years (60 months)

```
term <- 60
```

Model 1 - auto.arima

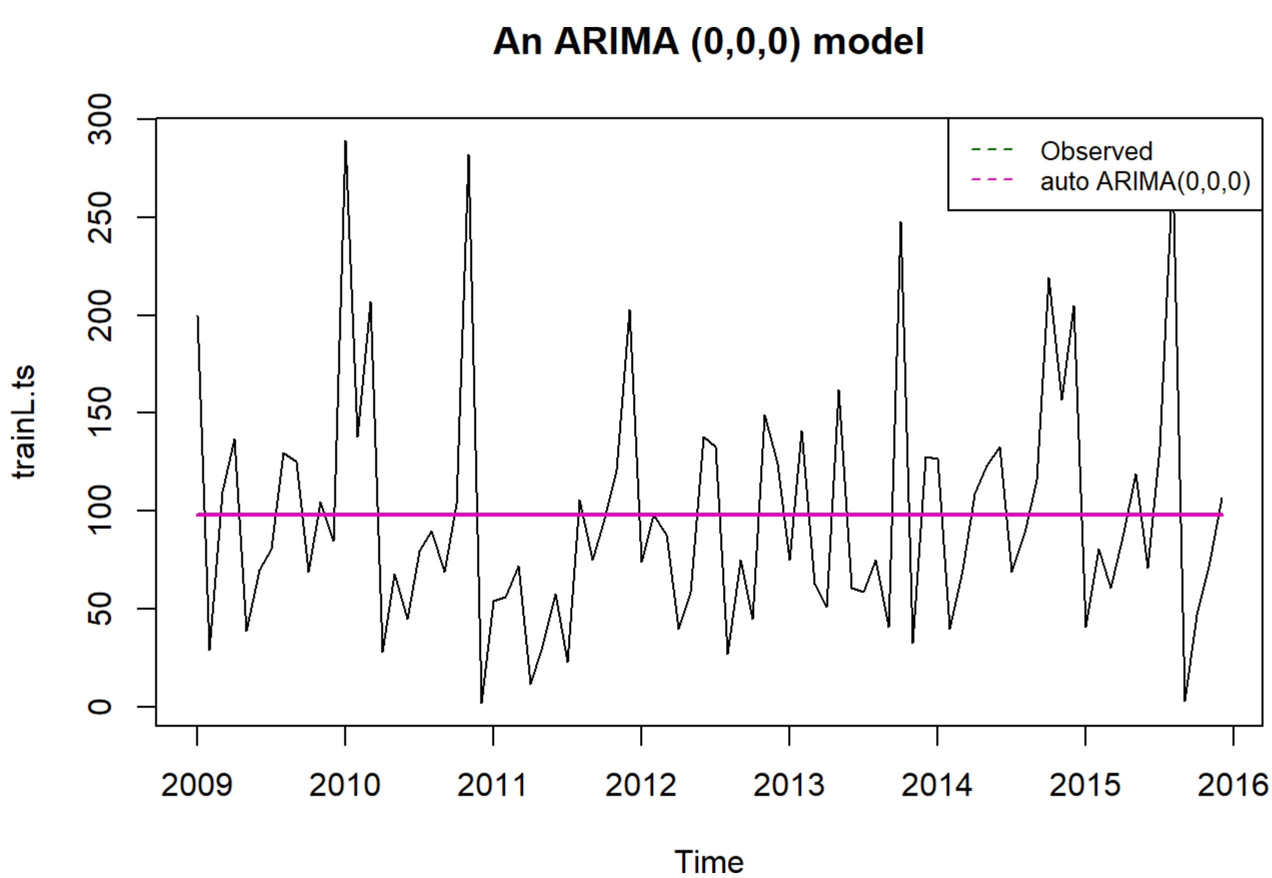
The first model auto.arima will present us with the best model with the lowest AIC.

```
# set seasonal = FALSE since there's no seasonal signals in our data series
autoarima.Model1 <- auto.arima(trainL.ts, ic = "aic", trace = TRUE, seasonal = FALSE, stepwise
 = FALSE)
```

```
##
##  ARIMA(0,0,0)           with zero mean     : 1038.405
##  ARIMA(0,0,0)           with non-zero mean : 933.8726
##  ARIMA(0,0,1)           with zero mean     : 1012.902
##  ARIMA(0,0,1)           with non-zero mean : 935.7065
##  ARIMA(0,0,2)           with zero mean     : 997.1395
##  ARIMA(0,0,2)           with non-zero mean : 937.6647
##  ARIMA(0,0,3)           with zero mean     : 991.0892
##  ARIMA(0,0,3)           with non-zero mean : 939.0451
##  ARIMA(0,0,4)           with zero mean     : 988.1105
##  ARIMA(0,0,4)           with non-zero mean : 940.1819
##  ARIMA(0,0,5)           with zero mean     : 984.7808
##  ARIMA(0,0,5)           with non-zero mean : 942.1556
##  ARIMA(1,0,0)           with zero mean     : 983.6068
##  ARIMA(1,0,0)           with non-zero mean : 935.6991
##  ARIMA(1,0,1)           with zero mean     : Inf
##  ARIMA(1,0,1)           with non-zero mean : 937.6131
##  ARIMA(1,0,2)           with zero mean     : Inf
##  ARIMA(1,0,2)           with non-zero mean : 939.5918
##  ARIMA(1,0,3)           with zero mean     : Inf
##  ARIMA(1,0,3)           with non-zero mean : 940.2665
##  ARIMA(1,0,4)           with zero mean     : Inf
##  ARIMA(1,0,4)           with non-zero mean : 942.1234
##  ARIMA(2,0,0)           with zero mean     : 966.9016
##  ARIMA(2,0,0)           with non-zero mean : 937.6491
##  ARIMA(2,0,1) with zero mean     : Inf
##  ARIMA(2,0,1)           with non-zero mean : 939.5988
##  ARIMA(2,0,2)           with zero mean     : Inf
##  ARIMA(2,0,2)           with non-zero mean : Inf
##  ARIMA(2,0,3)           with zero mean     : Inf
##  ARIMA(2,0,3)           with non-zero mean : 938.3046
##  ARIMA(3,0,0)           with zero mean     : 963.7751
##  ARIMA(3,0,0)           with non-zero mean : 939.2568
##  ARIMA(3,0,1)           with zero mean     : Inf
##  ARIMA(3,0,1)           with non-zero mean : 940.2337
##  ARIMA(3,0,2) with zero mean     : Inf
##  ARIMA(3,0,2)           with non-zero mean : Inf
##  ARIMA(4,0,0)           with zero mean     : 963.1107
##  ARIMA(4,0,0)           with non-zero mean : 940.2656
##  ARIMA(4,0,1)           with zero mean     : Inf
```

```
##   ARIMA(4,0,1)             with non-zero mean : 941.9804
##   ARIMA(5,0,0)             with zero mean     : 961.5915
##   ARIMA(5,0,0)             with non-zero mean : 942.1781
##
##
##
##   Best model: ARIMA(0,0,0)             with non-zero mean
```

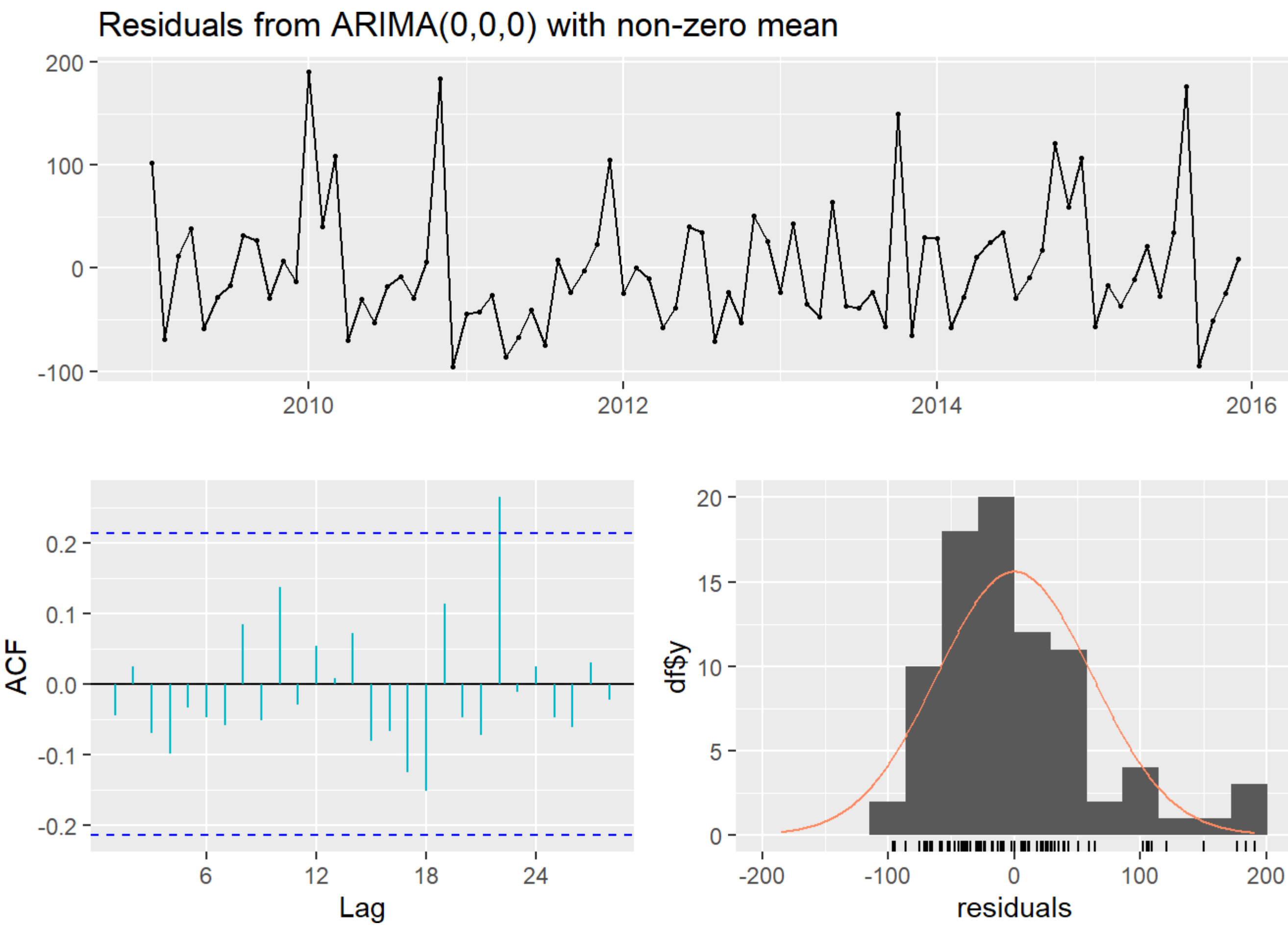
```
plot(trainL.ts, main = "An ARIMA (0,0,0) model")
lines(fitted(autoarima.Model1), col = "#dc0ab4", lwd = 2)
legend("topright", c("Observed", "auto ARIMA(0,0,0)"), lty = 8, col = c("darkgreen", "#dc0ab4"
), cex = 0.8)
```



An ARIMA(0,0,0) model is pretty flat.

Examine model 1 residuals

```
forecast::checkresiduals(autoarima.Model1, col = "#00B7C7")
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0) with non-zero mean
## Q* = 8.6454, df = 17, p-value = 0.9507
##
## Model df: 0.   Total lags used: 17
```

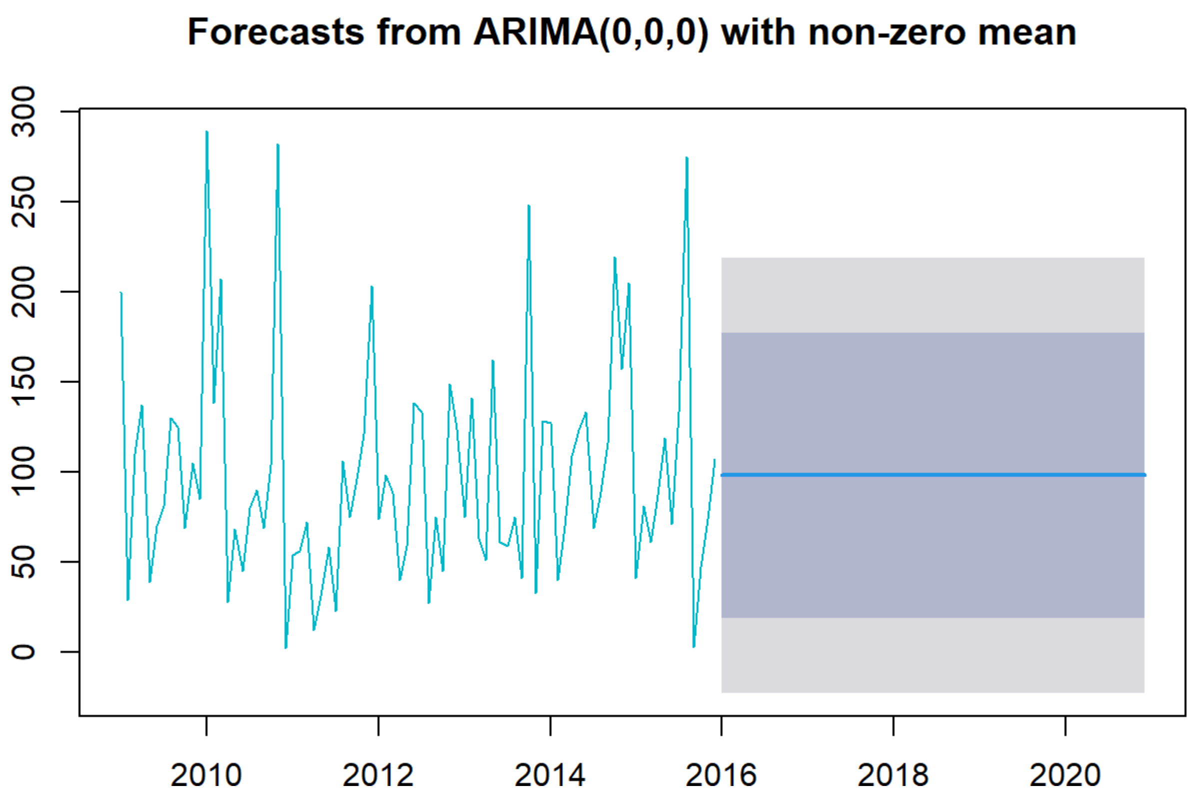
Observed graph: The first graph shows the residuals of the observed data series.\

ACF plot: The residuals of our first (auto.arima) model are not that autocorrelated which is good. There's only one peak, a lag on time step 22, that goes beyond the 95% limits of ACF values. We'll address the lag on the next model. Note that autocorrelation refers to a problem in data collected repeatedly over time.\

Residual histogram: The residuals doesn't quite follow a normal distribution, it has a couple of bins with very high concentration of cases and other low bins which distort the normal distribution.\

Initialize the forecast term to 60 months (5 years), forecast Model 1, and plot it.

```
# h is the forecast horizon value, set it to the defined term; otherwise it defaults to 2 year
s forecast.
autoarima.Model1.Fcast <- forecast(autoarima.Model1, h = term)
plot(autoarima.Model1.Fcast, col = "#00B7C7")
```



The plot shows observed and forecast data series, the prediction is just a flat line at

```
fcast.mean <- autoarima.Model1.Fcast$mean[1:1]
formattable(fcast.mean, digits = 2, format = "f")
```

```
## [1] 98.00
```

It's a worthy to note about these two terms:\ fcast$fitted is the result of the fit (the model fitted to observation)\

fcast$mean is the result of the forecast (the application of the model to the future).\

These two terms have a different length for a given h.

Check how well Model 1 forecast

```r
# Check how accurate the forecast is
autoarima.Model1.Fcast.em <- forecast(autoarima.Model1, h = term) %>%
  accuracy(validL.ts)

# Evaluate TS forecast with regression evaluation metrics:
round(autoarima.Model1.Fcast.em[, c("RMSE", "MAPE")], 2)
```

```
##                 RMSE   MAPE
## Training set 61.31 161.45
## Test set     60.54 301.40
```

Examine Model 1 coefficients

```
## Series: trainL.ts
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
##        mean
##       98.00
## s.e.   6.69
##
## sigma^2 = 3805:  log likelihood = -464.94
## AIC=933.87   AICc=934.02   BIC=938.73
##
## Training  set error measures:
##                        ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 1.353617e-14 61.31457 46.92857 -136.1746 161.4497 0.7166187
##                     ACF1
## Training set -0.04492774
```

Akaike information criteria (AIC) is a mathematical method for evaluating how well a model fits the data it was generated from.\ AIC shows us how good a model is relative to the other models.\ Root mean square error (RMSE) tells us how many units our model is wrong on average.\ Mean absolute percentage error (MAPE) tells us how wrong our forecasts are percentage-wise.\ The lower the AIC/RMSE the better the model, likewise, the lower the MAPE the more accurate the forecast is. \

We'll keep track of AIC and RMSE and store them in an error measure (em) table for comparison with other models as we progressively fit.

```
# Format the coefficient into an integer
model1.AIC <- formattable(stats::AIC(autoarima.Model1), digits = 1, format = "f")
model1.RMSE <- formattable(autoarima.Model1.Fcast.em[1, c("RMSE")], digits = 1, format = "f")

# rm(em_results)
em_results <- tibble(
  Method = "Model 1 - auto.arima ARIMA(0,0,0)",
  AIC = model1.AIC,
  RMSE = model1.RMSE
)
em_results %>%
  kbl(caption = "Models Performance Table") %>%
  kable_classic_2(full_width = F, c("striped", "hover"))
```

Models Performance Table

Method	AIC	RMSE
Model 1 - auto.arima ARIMA(0,0,0)	933.9	61.3

Model 2 - ARIMA(0,0,1)

Previously in the ACF plot on figure ** Residuals from ARIMA(0,0,0) ** shows a spike at lag 22 but no other significant spikes; this suggests that the model may better with a different specification, such as p=22 or q=22.

ARIMA can be identified as the order of AR, I, MA terms. An ARIMA model has three component functions: The order of the non-seasonal auto-regressive (AR) terms. If p = NULL, an optimal number of lags will be selected for a linear AR(p) model via AIC. I(d) is the difference in the nonseasonal observations; and MA(q) is the size of the moving average window.

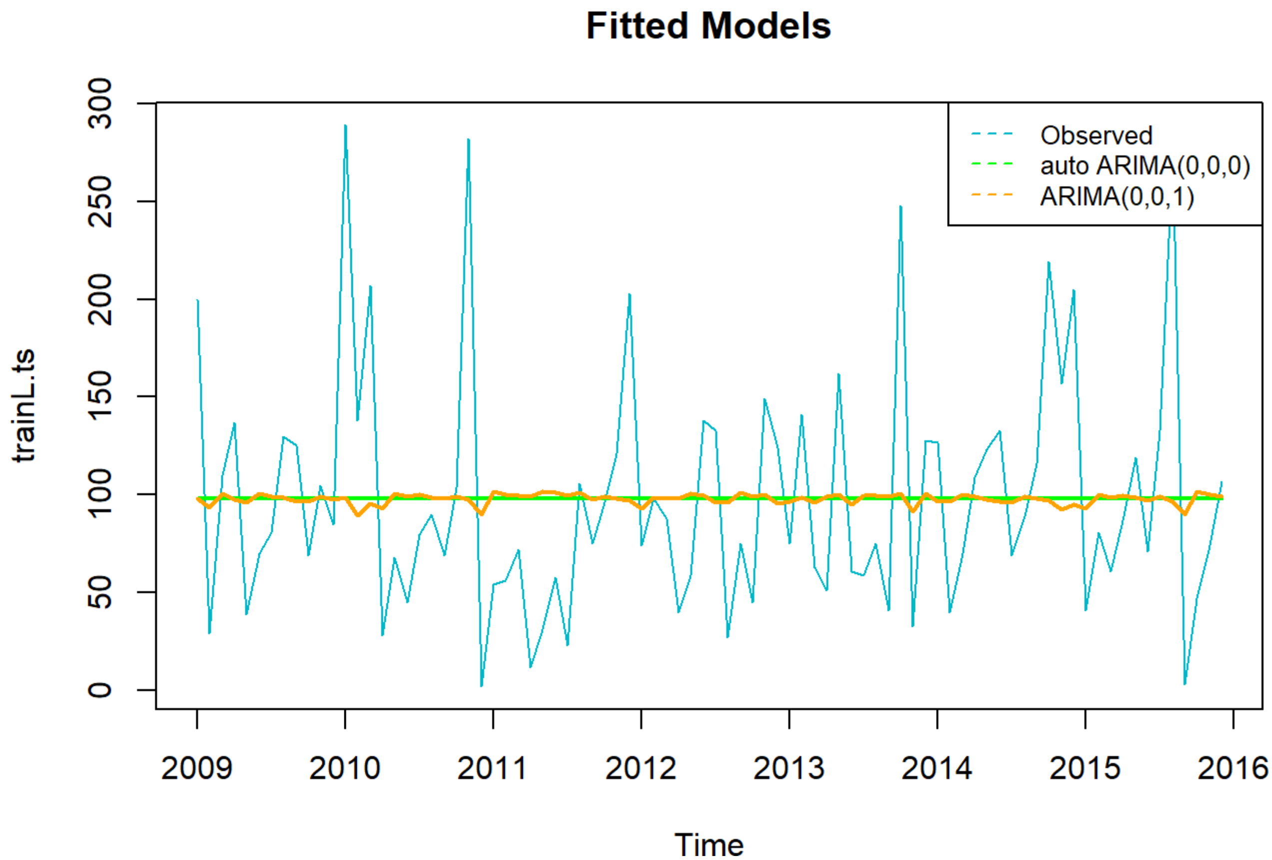
ARIMA (0,0,22) was fitted and evaluated; There was a noticeably huge difference in the RMSE between the two data sets. The model may had been overfitted.

Training set 50.04 and Test set 60.66.

The model was modified from ARIMA (0,0,22) to ARIMA(0,0,1).

For the second model, we identify AR = 0, I=0, and MA=1 or simply called it an ARIMA model for a first order of MA process. We can repeat the fitting process allowing for the MA(1) component and examine diagnostic and plot.

```
MA1.model2 <- forecast::Arima(trainL.ts, c(0, 0, 1))
plot(trainL.ts, col = "#00B7C7", main = "Fitted Models")
lines(fitted(autoarima.Model1), col = "green", lwd = 2)
lines(fitted(MA1.model2), col = "#ffa300", lwd = 2)
legend("topright", c("Observed", "auto ARIMA(0,0,0)", "ARIMA(0,0,1)"), lty = 8, col = c("#00B7
C7", "green", "#FFA300"), cex = 0.8)
```

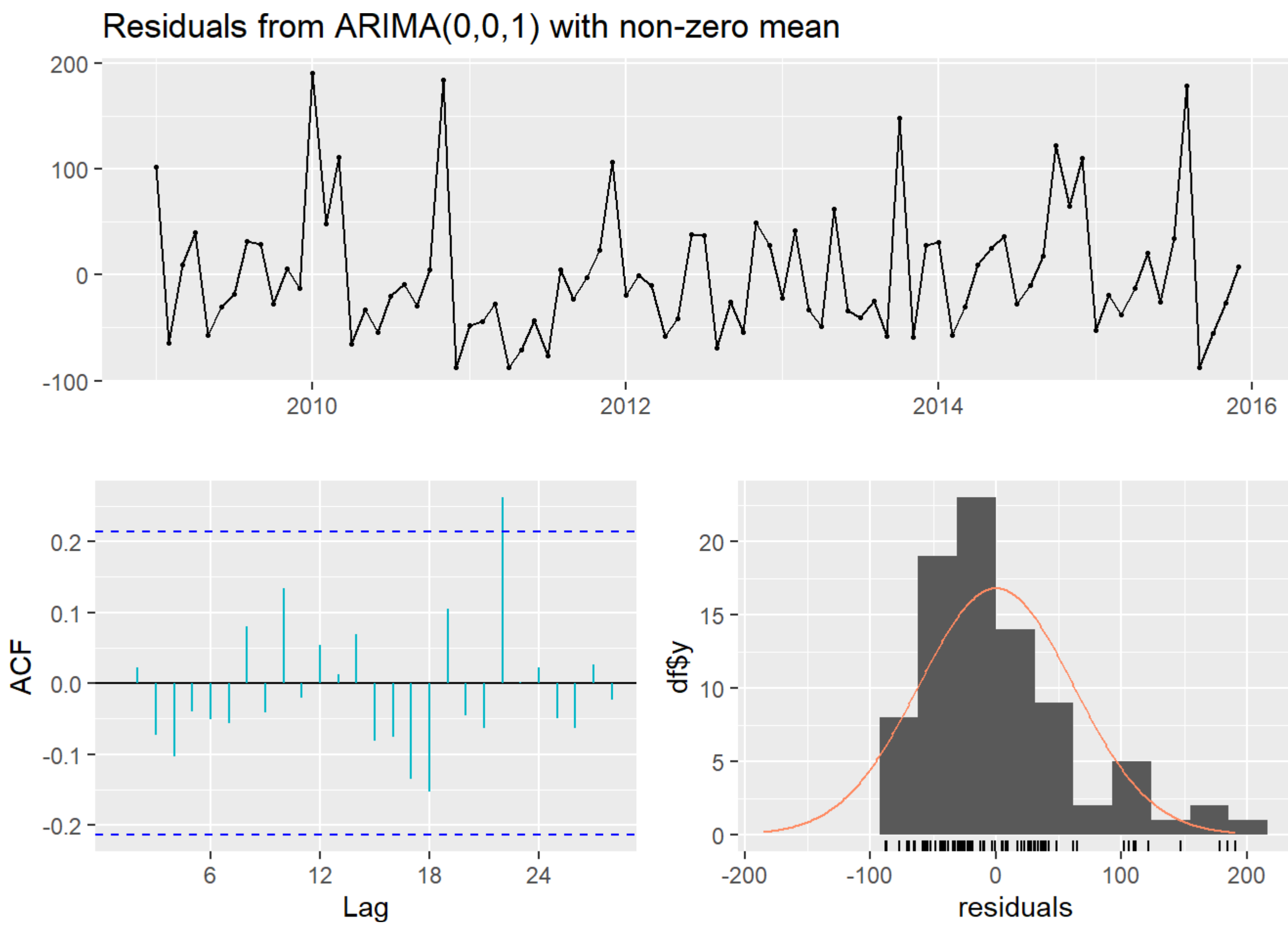


Visually Model 1 and 2 look very similar. Let's explore how model 2 is fitting.

```
## Series: trainL.ts
## ARIMA(0,0,1) with non-zero mean
##
## Coefficients:
##           ma1     mean
##       -0.0441  97.9440
## s.e.   0.1083   6.3936
##
## sigma^2 = 3843:  log likelihood = -464.85
## AIC=935.71   AICc=936.01   BIC=943
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.05296268 61.25327 46.99669 -128.2694 153.5474 0.7176589
##                      ACF1
## Training set 2.937896e-05
```

Model 2 residuals plots

```
forecast::checkresiduals(MA1.model2, col = "#00B7C7")
```

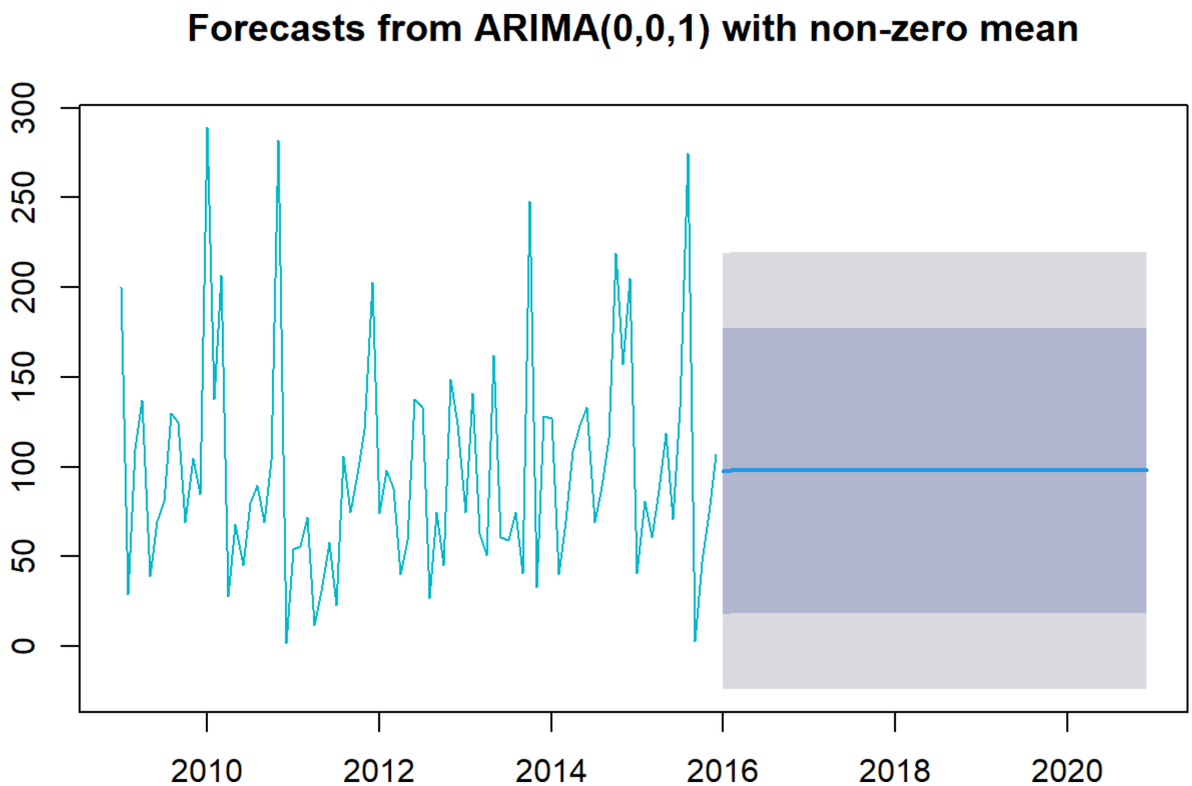



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,1) with non-zero mean
## Q* = 8.7871, df = 16, p-value = 0.9219
##
## Model df: 1.   Total lags used: 17
```

Observed graph: The residuals of the observed data. **ACF plot:** There is a spike at time step 22 and everything else seems to be within acceptable range.
Residual histogram: The residuals still doesn't follow a normal distribution, it has a couple of bins with very high concentration of live cancer cases then cascade down to the other lower bins on the right which distort the normal distribution.

Forecast from Model 2

```
MA1.model2.Fcast <- forecast(MA1.model2, h = term)
plot(MA1.model2.Fcast, col = "#00B7C7")
```



Model 2 forecast shows a flat lined prediction at

```
## [1] 97.6
```

Check how well Model 2 forecast

```
# Evaluate TS forecast with regression evaluation metrics:
# Check how accurate the forecast is
MA1.model2.Fcast.em <- forecast(MA1.model2, h = term) %>%
  accuracy(validL.ts)

# Check TS forecast accuracy with regression evaluation metrics:
MA1.model2.Fcast.em[, c("RMSE", "MAPE")]
```

```
##                  RMSE     MAPE
## Training set 61.25327 153.5474
## Test set     60.52440 301.1880
```

Record our findings.

Models Performance Table

Method	AIC	RMSE
Model 1 - auto.arima ARIMA(0,0,0)	933.9	61.3
Model 2 - ARIMA(0,0,1)	935.7	61.3

AIC measures how well the model will fit new data, not the existing data. Lower AIC means that a model should have improved prediction. Frequently adding more variables decreases predictive accuracy and in that case the model with higher RMSE will have a higher (worse) AIC.

The AIC quantifies the goodness of fit and simplicity of the model into a single statistic. When comparing two models, the one with the lower AIC is considered to be better; however, the RMSE is a frequently used measure of the differences between values predicted by a model or an estimator and the values observed. The lower the RMSE the better when calculating the accuracy of predictions of a model. (Tracyenee 2022)

Even though both AIC and RMSE are being tracked, the model with the lowest RMSE will be selected due to the objective of this project, accurate forecasting.

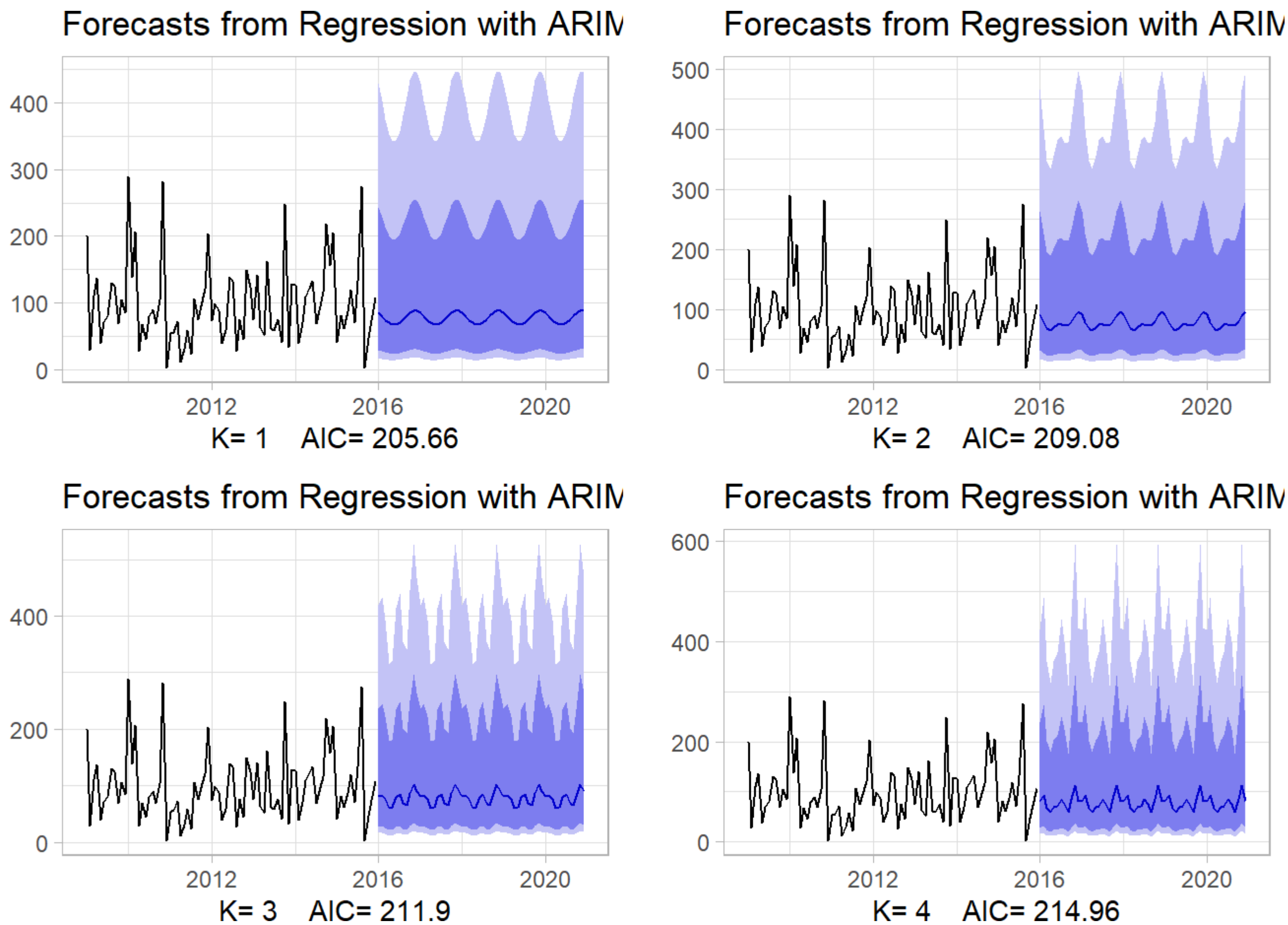
Model 3 - ARIMA(0,0,0) with Fourier Term

Using an ARIMA model alone does not sufficiently capture the long-term patterns, the Fourier term is introduced into the model.
Ludlow & Enders (2000, IJF)

K - every periodic function can be approximated by sums of sin and cos terms for large enough K. The best way to select K is to try a few different values and select the model that gives the lowest AIC values. Choose K to minimize the AIC start with K =1 and slowly increase it until the AICs value stops decreasing.

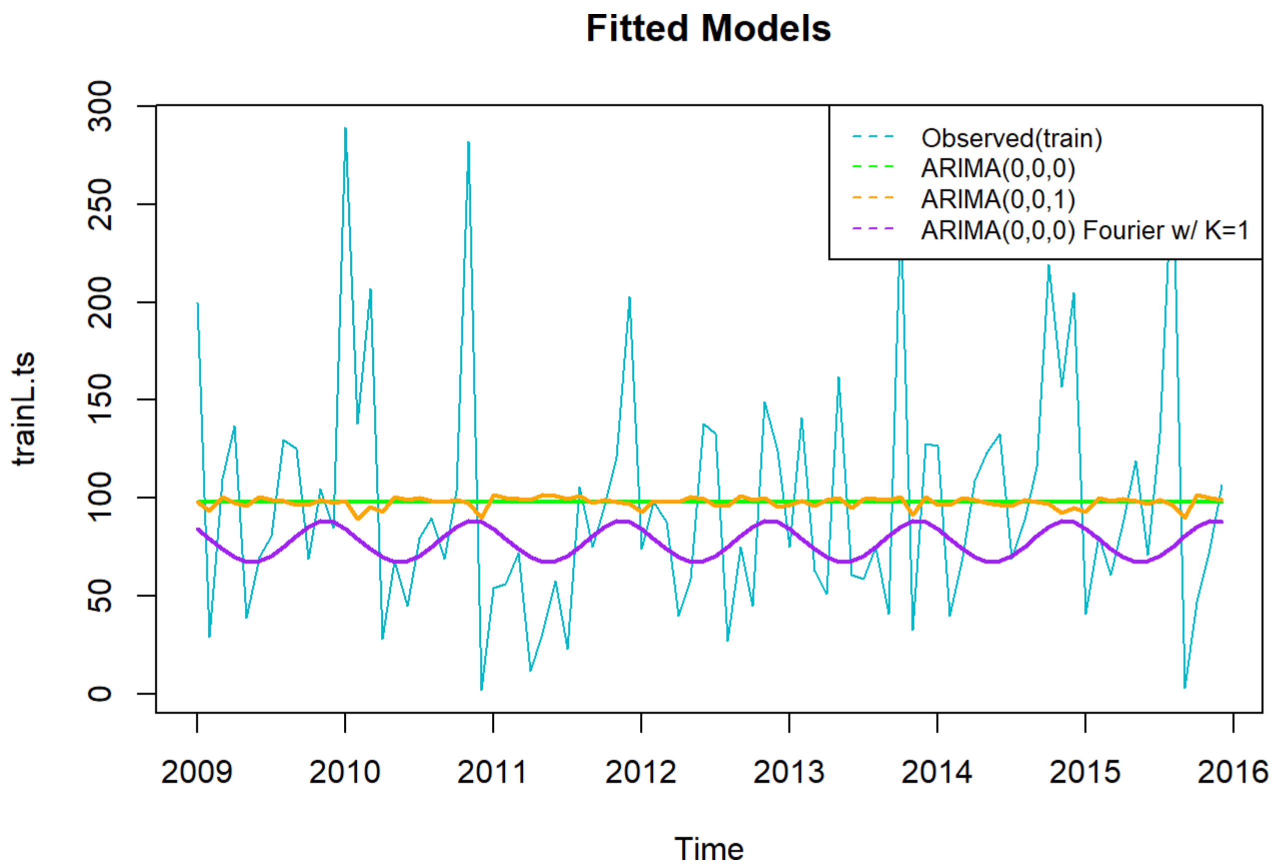
Check which K term is best for our 4th model

```
###########################################
# Model 3
# Approaches to TS data with weak seasonality.
##########################################
# Comparing with plots
plots <- list()
for (i in seq(4)) {
  fit <- trainL.ts %>%
    auto.arima(xreg = fourier(trainL.ts, K = i), seasonal = FALSE, lambda = "auto")
  plots[[i]] <- autoplot(forecast(fit, xreg = fourier(trainL.ts, K = i, h = term))) +
    xlab(paste("K=", i, "    AIC=", round(fit[["aic"]], 2))) +
    ylab("") +
    theme_light()
}

gridExtra::grid.arrange(
  plots[[1]], plots[[2]],
  plots[[3]], plots[[4]],
  nrow = 2
)
```



It seems K=1 has the lowest AIC value. Fit model 3 with K=1 and plot it with the other fitted models.

```
# Modeling with Fourier Regression
fit.fourier.model3 <- trainL.ts %>%
  auto.arima(xreg = fourier(trainL.ts, K = 1), seasonal = FALSE, lambda = "auto")

# Plot fitted models
plot(trainL.ts, col = "#00B7C7", main = "Fitted Models")
lines(fitted(autoarima.Model1), col = "green", lwd = 2)
lines(fitted(MA1.model2), col = "#ffa300", lwd = 2)
lines(fitted(fit.fourier.model3), col = "purple", lwd = 2)
legend("topright", c("Observed(train)", "ARIMA(0,0,0)", "ARIMA(0,0,1)", "ARIMA(0,0,0) Fourier
w/ K=1"), lty = 8, col = c("#00B7C7", "green", "#FFA300", "purple"), cex = 0.8)
```

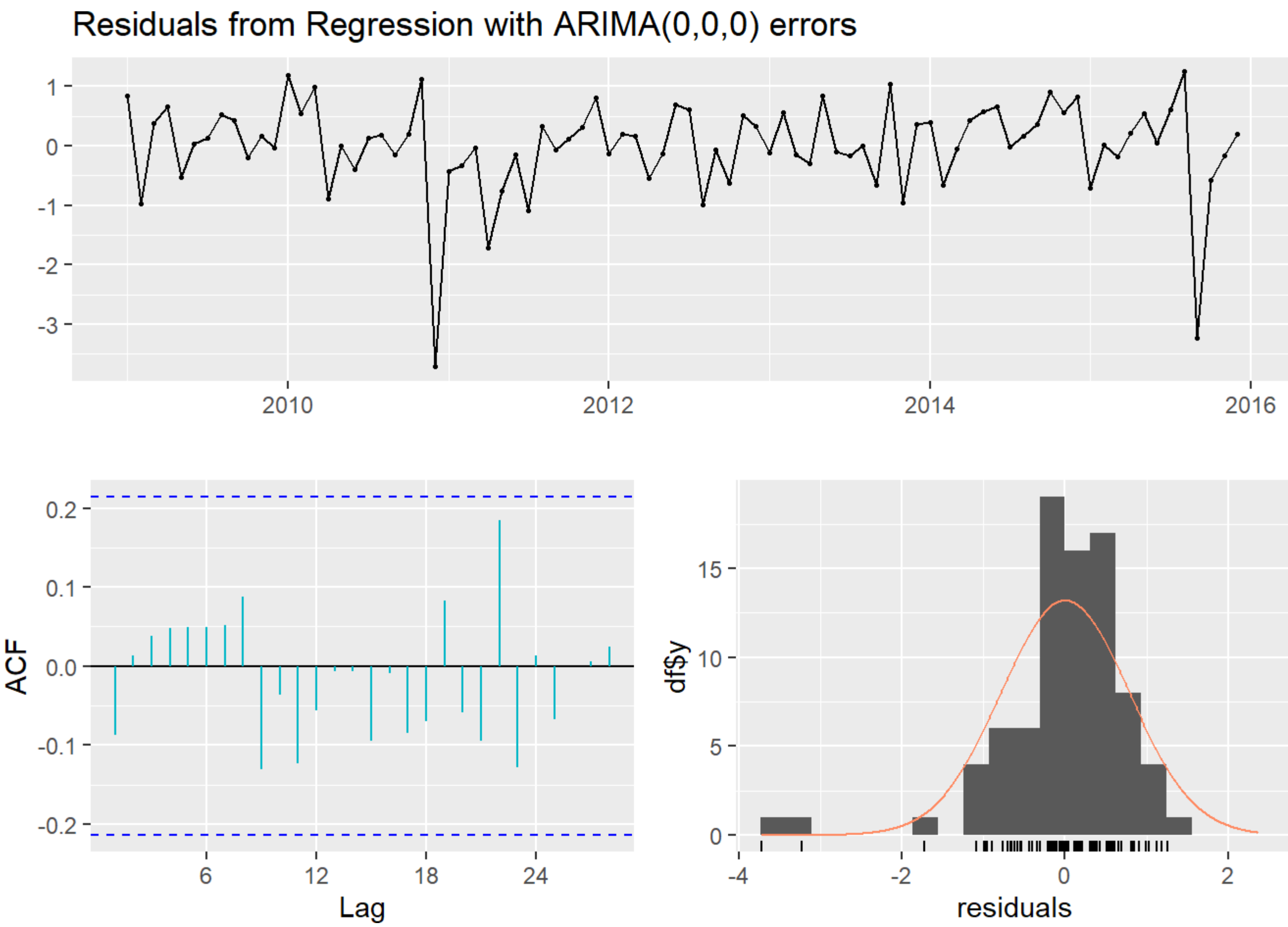



```
summary(fit.fourier.model3)
```

```
## Series: .
## Regression with ARIMA(0,0,0) errors
## Box Cox transformation: lambda= -0.006889242
##
## Coefficients:
##         intercept    S1-12    C1-12
##            4.2826  -0.0410   0.1255
## s.e.       0.0856   0.1211   0.1211
##
## sigma^2 = 0.6386:  log likelihood = -98.83
## AIC=205.66   AICc=206.17   BIC=215.38
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 20.38854 63.18001 44.25446 -92.12438 131.2875 0.6757839
##                     ACF1
## Training set -0.08790084
```

Noticeably drop of Model 3 AIC value

```
forecast::checkresiduals(fit.fourier.model3, col = "#00B7C7")
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(0,0,0) errors
## Q* = 7.7345, df = 17, p-value = 0.9719
##
## Model df: 0.   Total lags used: 17
```

ACF plot: The residuals of Model 3 seem to be within acceptable range.
Residual histogram: The residuals doesn't quite follow a normal distribution, it has bins with very high concentration of cases then a couple of trail off lower bins on the left which again distort the normal distribution.

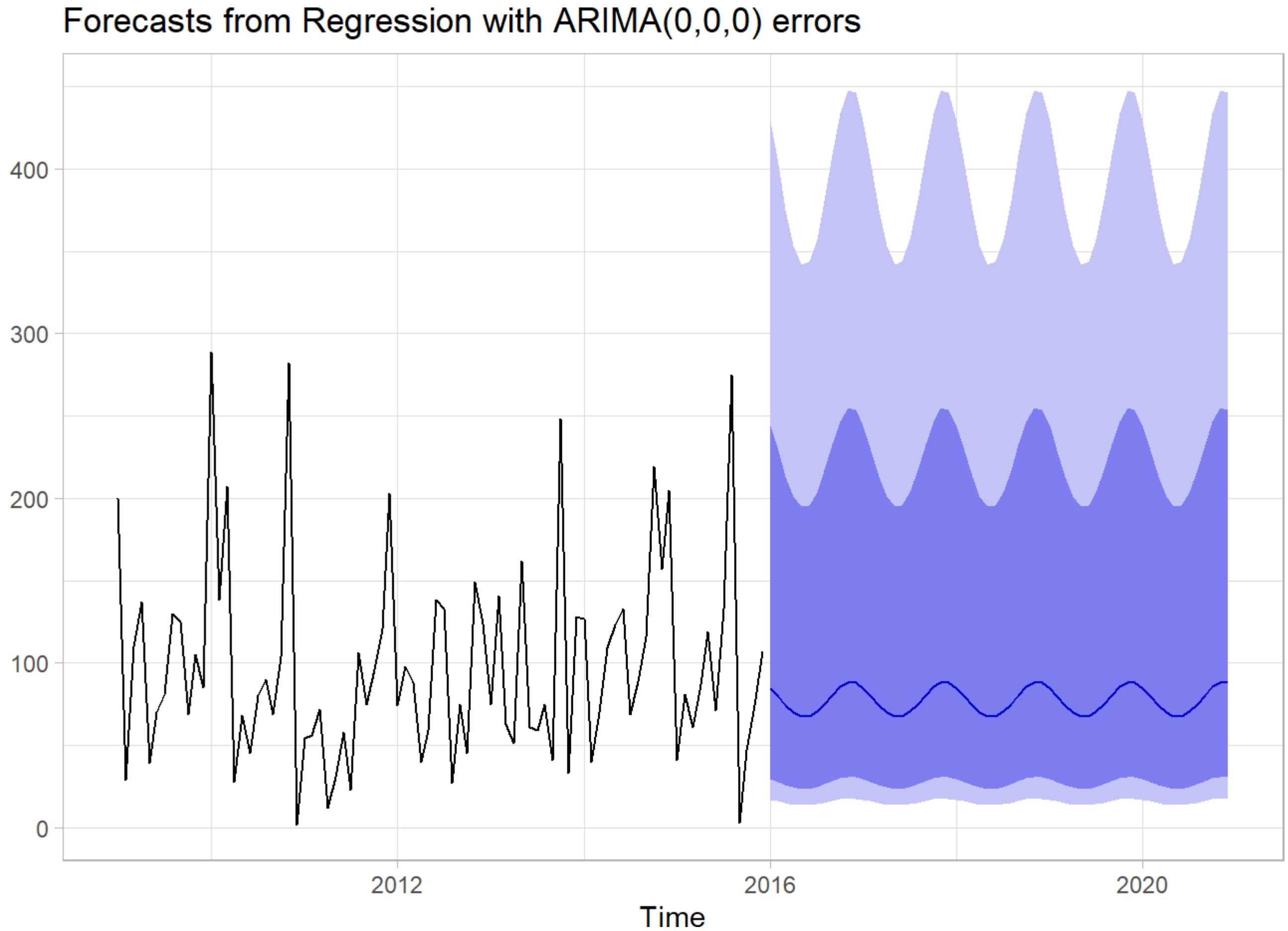
Check how well our 3rd fitted model fair between training and test set (Validation).

```
##               RMSE  MAPE
## Training set 63.2 131.3
## Test set     63.0 244.4
```

The results look very compatible between the two data sets. Plot Model 3 Fourier Regression forecast

```
# Plot of the Fourier Regression Model 3 forecast, train.ts fit and valid.ts
fit.fourier.model3.fcast <- forecast(fit.fourier.model3, xreg = fourier(trainL.ts, K = 1, h = term))

autoplot(fit.fourier.model3.fcast) +
```

```
  theme_light() +
  ylab("")
```



Our data don't have any trend or seasonality; however this forecast and predicted data (below) seems to tell a different story.

```
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 2016 84.6 79.5 74.1 69.8 67.6 67.9 70.5 75.1 80.6 85.5 88.3 87.9
## 2017 84.6 79.5 74.1 69.8 67.6 67.9 70.5 75.1 80.6 85.5 88.3 87.9
## 2018 84.6 79.5 74.1 69.8 67.6 67.9 70.5 75.1 80.6 85.5 88.3 87.9
## 2019 84.6 79.5 74.1 69.8 67.6 67.9 70.5 75.1 80.6 85.5 88.3 87.9
## 2020 84.6 79.5 74.1 69.8 67.6 67.9 70.5 75.1 80.6 85.5 88.3 87.9
```

Record Model 3 performance

Models Performance Table

Method	AIC	RMSE
Model 1 - auto.arima ARIMA(0,0,0)	933.9	61.3
Model 2 - ARIMA(0,0,1)	935.7	61.3
Model 3 - ARIMA(0,0,0) with Fourier K=1	205.7	63.2

Based on AIC value, Model 3 seems to lead.

Model 4 - ARIMA(0,0,0) with Transformed Data

Setting approximation = FALSE makes auto.arima work harder to find the right solution. Box Cox transformations help determine what is the best way to transform your data based on the lambda. Lambda here is used to represent the number that will be used to select the optimal transformation for the data. The optimal transformation of the data is that transformation that makes the data approximate the most to a normal distribution.

These two other methods allow for constants to be added to the model and for more complex models to be considered.
Drift: Only available when the differencing is above 0 and allows models with a changing average to be fit.
Mean: Allows models with a non-zero mean to be considered.

By default, R sets them as TRUE, again opting for speed over performance. Setting these parameters to FALSE allows the model to work harder, but watch out for overfitting. (Losada 2020)

```
fit.arima.trans.model4 <- trainL.ts %>%
  auto.arima(stepwise = FALSE, approximation = FALSE, lambda = "auto")
fit.arima.trans.model4
```

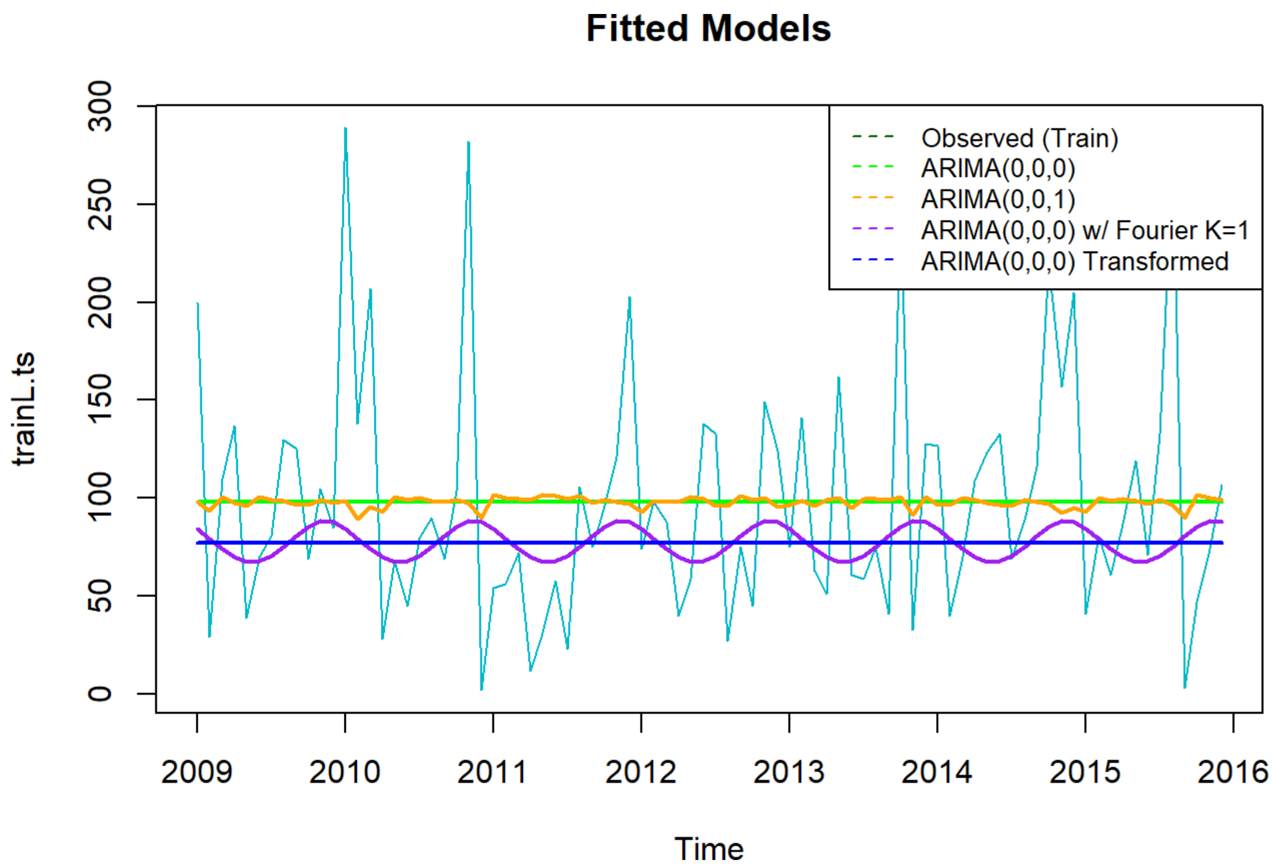
```
## Series: .
## ARIMA(0,0,0) with non-zero mean
## Box Cox transformation: lambda= -0.006889242
##
## Coefficients:
##          mean
##        4.2826
## s.e.  0.0862
##
## sigma^2 = 0.6321:  log likelihood = -99.42
## AIC=202.84   AICc=202.99   BIC=207.7
```

As seen above code chunk, stepwise=FALSE, approximation=FALSE parameters are used to amplify the searching for all possible model options. We set lambda parameter to “auto”. It makers the data transformed with lambda= -0.007.

From the results above ARIMAR(0,0,0) which can be denoted as ARIMA(p,d,q) we can see that there is no autoregressive (AR) part of the model, order moving average (MA), or differencing (I).

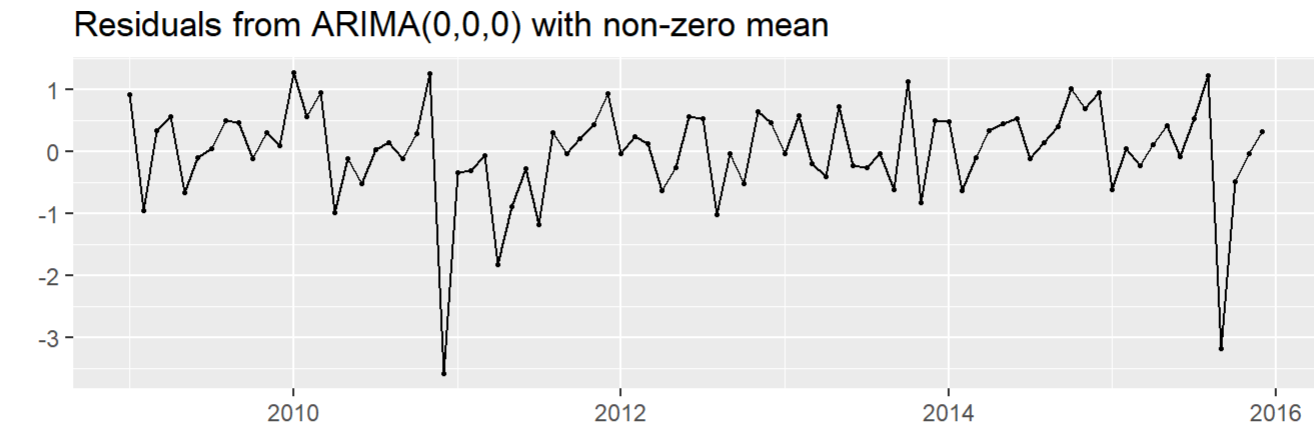
Based on the AIC, this model seems to fitted better than the previous models.

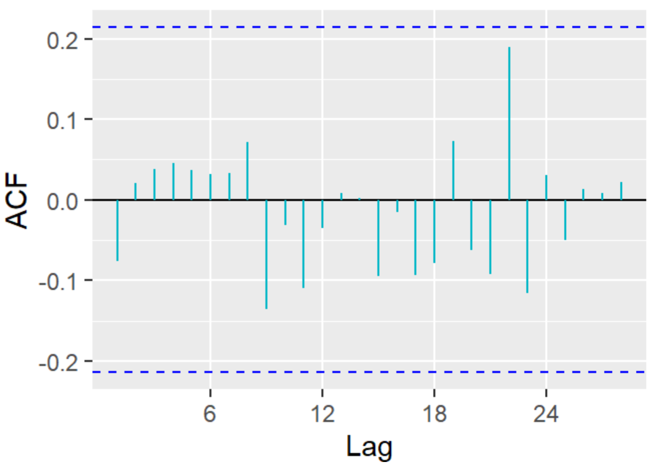
```
# Plot fitted models
plot(trainL.ts, col = "#00B7C7", main = "Fitted Models")
lines(fitted(autoarima.Model1), col = "green", lwd = 2)
lines(fitted(MA1.model2), col = "#ffa300", lwd = 2)
lines(fitted(fit.fourier.model3), col = "purple", lwd = 2)
lines(fitted(fit.arima.trans.model4), col = "blue", lwd = 2)
legend("topright", c("Observed (Train)", "ARIMA(0,0,0)", "ARIMA(0,0,1)", "ARIMA(0,0,0) w/ Four
ier K=1", "ARIMA(0,0,0) Transformed"), lty = 8, col = c("darkgreen", "green", "#FFA300", "purp
le", "blue"), cex = 0.8)
```

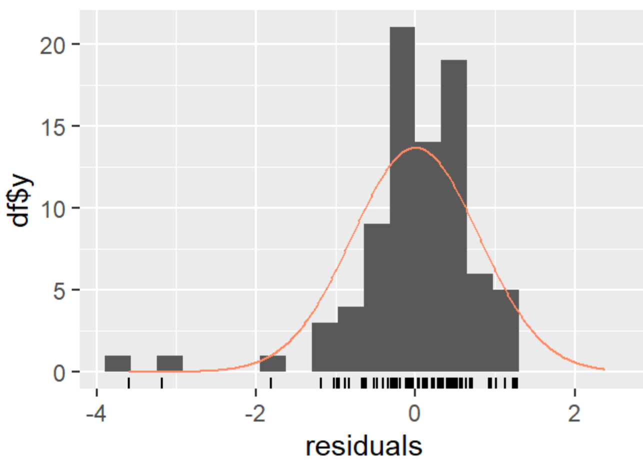



```
## Series: .
## ARIMA(0,0,0) with non-zero mean
## Box Cox transformation: lambda= -0.006889242
##
## Coefficients:
##          mean
##        4.2826
## s.e.  0.0862
##
## sigma^2 = 0.6321:  log likelihood = -99.42
## AIC=202.84   AICc=202.99   BIC=207.7
##
## Training set error measures:
##                    ME     RMSE      MAE     MPE     MAPE      MASE        ACF1
## Training set 20.74885 64.73014 45.41661 -86.171 125.9319 0.6935304 -0.04492774
```

Look how low the AIC is for Model 4!





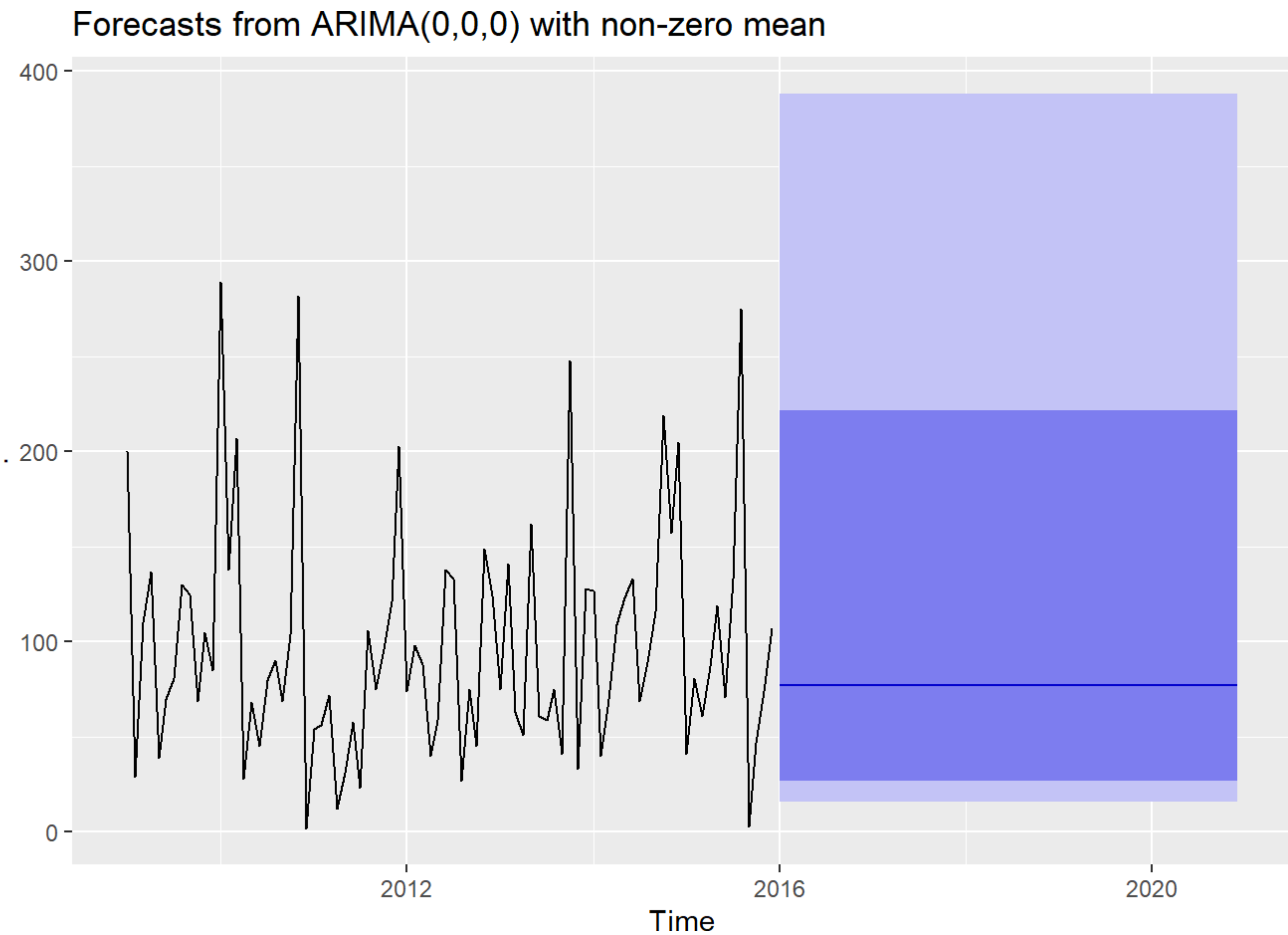



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,0) with non-zero mean
## Q* = 6.7373, df = 17, p-value = 0.9867
##
## Model df: 0.   Total lags used: 17
```

There is no lag indication in the ACF plot and residual histogram has slightly improved compared to previous model 3.

Forecast on Model 4

```
par(mfrow = c(1, 1))
fit.arima.trans.model4.fcast <- forecast(fit.arima.trans.model4, h = term)
autoplot(fit.arima.trans.model4.fcast)
```



With a non-seasonality, it's not uncommon to have a flat prediction.

```
## [1] 77.3
```

```
##                    RMSE     MAPE
## Training set 64.73014 125.9319
## Test set     61.79506 234.1500
```

Record Model 4 AIC

Models Performance Table

Method	AIC	RMSE
Model 1 - auto.arima ARIMA(0,0,0)	933.9	61.3
Model 2 - ARIMA(0,0,1)	935.7	61.3
Model 3 - ARIMA(0,0,0) with Fourier K=1	205.7	63.2
Model 4 - ARIMA(0,0,0) w/ Transformation	202.8	64.7

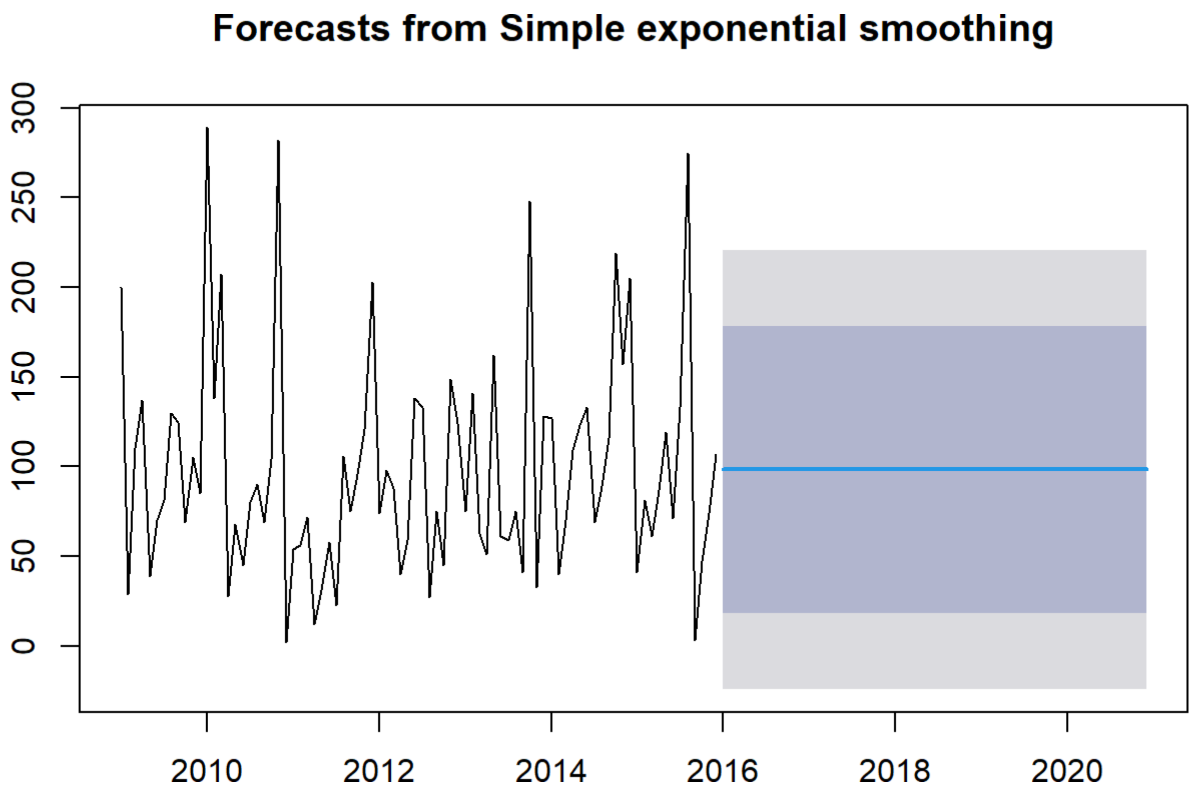
Model 5 - Single Exponential Smoothing (SES)

Single Exponential Smoothing (SES) is useful for forecasting a series with no trend and no seasonality. SES forecasts future values using a weighted average of all previous values in the series. Advantages of this method is that it's simple, popular, and adaptive. The key concepts is smoothing constant. This method, which results in a straight, flat-line forecast is best for volatile data with no trend or seasonality. (GreeksforGeeks 2022)
Start Model 5a with a smaller alpha = 0.01; fit & forecast the model, and examine its coefficients

```
ses.fit.model5a <- ses(trainL.ts,
  alpha = 0.01,
  h = term
)

ses.fit.model5a.coef <- summary(ses.fit.model5a)
ses.fit.model5a.coef$model
```

```
## Simple exponential smoothing
##
## Call:
##  ses(y = trainL.ts, h = term, alpha = 0.01)
##
##   Smoothing parameters:
##     alpha = 0.01
##
##   Initial states:
##     l = 98.0139
##
##   sigma:  62.3491
##
##      AIC     AICc      BIC
## 1068.466 1068.614 1073.328
```

```
plot(ses.fit.model5a)
```



Model 5 flattens at

```
## [1] 98.3
```

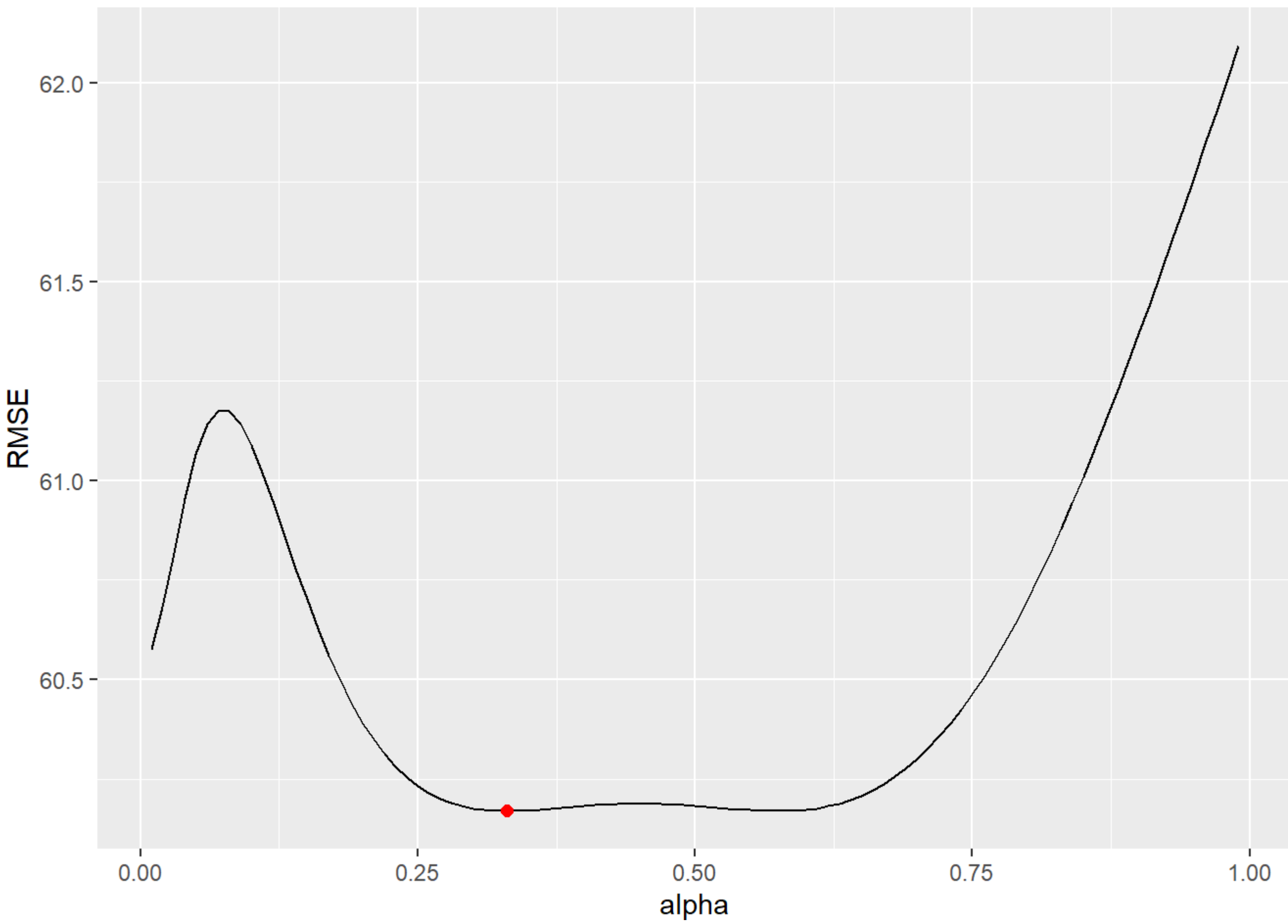
Model 5 accuracy

```
##              RMSE  MAPE
## Training set 61.6 164.2
## Test set     60.6 302.6
```

Compare models based on the lowest alpha

```
alpha <- seq(.01, .99, by = .01)
RMSE <- NA
for (i in seq_along(alpha)) {
  fit <- ses(trainL.ts,
    alpha = alpha[i],
    h = term
  )

  RMSE[i] <- accuracy(fit, validL.ts)[2, 2]
}

# convert to a data frame and identify min alpha value
alpha.fit <- tibble(alpha, RMSE)
# alpha.fit
```

```
alpha.min <- filter(
  alpha.fit,
  RMSE == min(RMSE)
)

ggplot(alpha.fit, aes(alpha, RMSE)) +
  geom_line() +
  geom_point(
    data = alpha.min,
    aes(alpha, RMSE),
    lwd = 2, color = "red"
  )
```




```
alpha.min
```

```
## # A tibble: 1 × 2
##   alpha  RMSE
##   <dbl> <dbl>
## 1  0.33  60.2
```

Now, we will try to re-fit our forecast model for SES with alpha = 0.33. We will notice the significant difference between alpha 0.01 and alpha=0.33.

```
ses.fit.model5b <- ses(trainL.ts,
  alpha = 0.33,
```

```
  h = term
)

ses.fit.model5b.coef <- summary(ses.fit.model5b)
ses.fit.model5b.coef$model
```

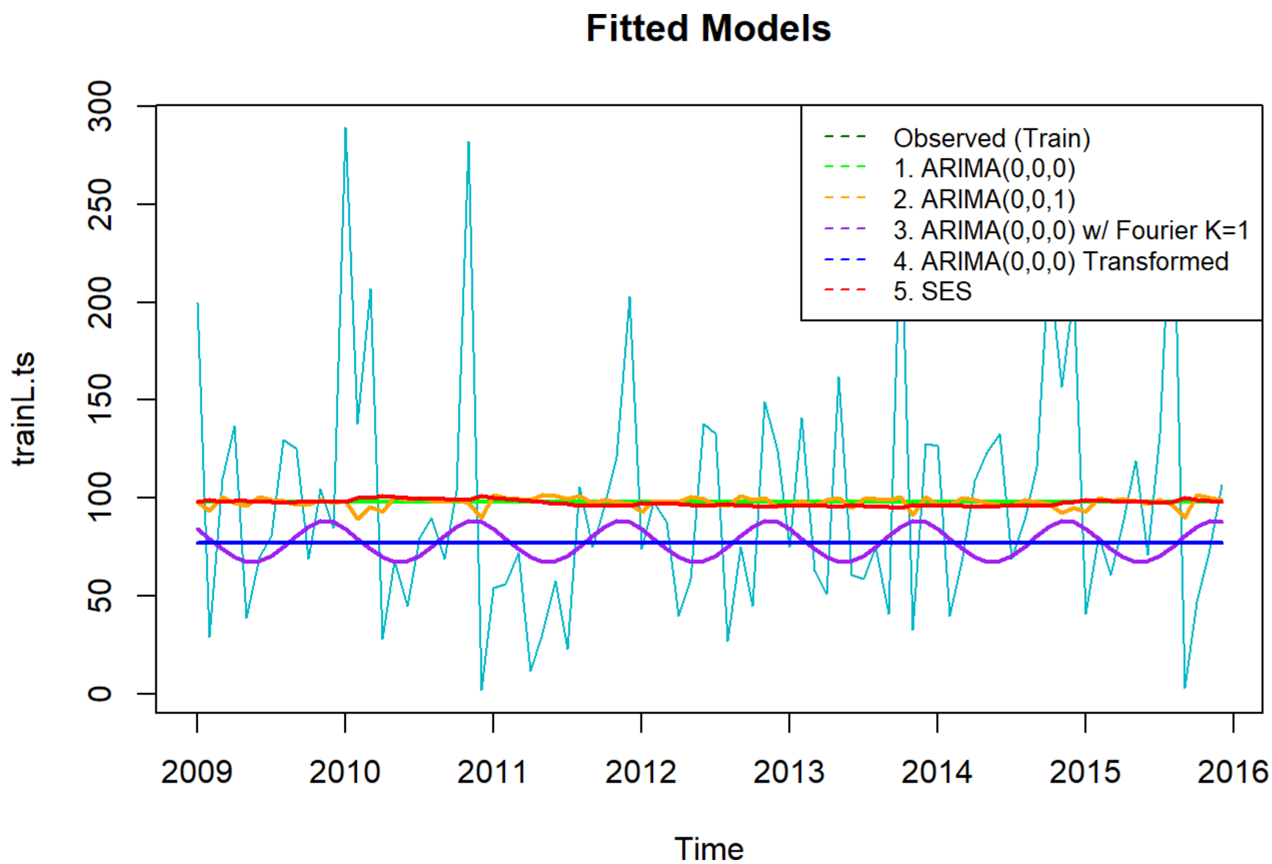
```
## Simple exponential smoothing
##
## Call:
##  ses(y = trainL.ts, h = term, alpha = 0.33)
##
##   Smoothing parameters:
##     alpha = 0.33
##
##   Initial states:
##     l = 117.7172
##
##   sigma:  68.9569
##
##      AIC     AICc      BIC
## 1085.389 1085.538 1090.251
```

Check model5b forecast accuracy

```
##                  RMSE     MAPE
## Training set 68.13105 221.8442
## Test set     60.16912 279.1185
```

Based on both AIC and RMSE, ses.fit.model5a does much better than ses.fit.model5b, we'll keep ses.fit.model5a as Model 5. Plot fitted models

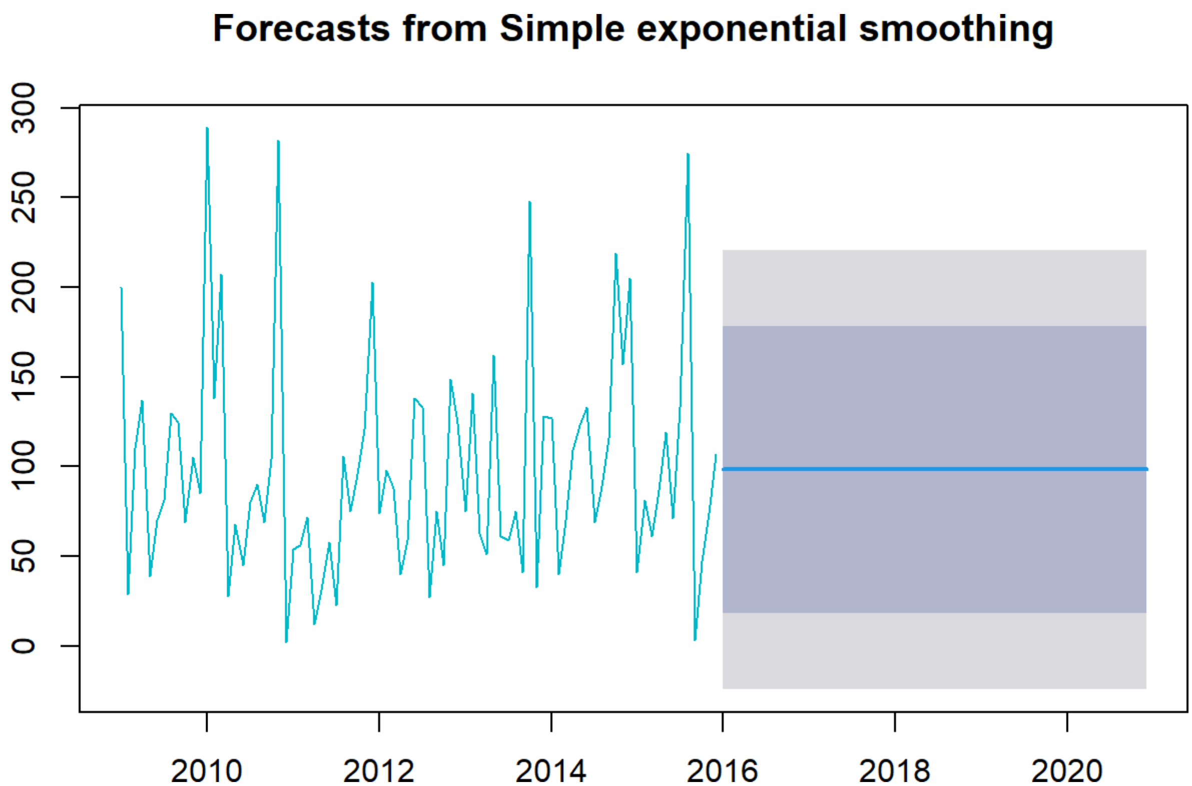
```
plot(trainL.ts, col = "#00B7C7", main = "Fitted Models")
lines(fitted(autoarima.Model1), col = "green", lwd = 2)
lines(fitted(MA1.model2), col = "#ffa300", lwd = 2)
lines(fitted(fit.fourier.model3), col = "purple", lwd = 2)
lines(fitted(fit.arima.trans.model4), col = "blue", lwd = 2)
lines(fitted(ses.fit.model5a), col = "red", lwd = 2)
legend("topright", c("Observed (Train)", "1. ARIMA(0,0,0)", "2. ARIMA(0,0,1)", "3. ARIMA(0,0,0
) w/ Fourier K=1", "4. ARIMA(0,0,0) Transformed", "5. SES"), lty = 8, col = c("darkgreen", "gr
een", "#FFA300", "purple", "blue", "red"), cex = 0.8)
```



Visually Model 1, 2, and 5 look closely similar.
Model 4 seems to be the average line running through Model 3.

Plot model 5 forecast

```
plot(ses.fit.model5a, col = "#00B7C7")
```



Flat Prediction at

```
## [1] 98.3
```

Examine Model 5 AIC

```
ses.fit.model5a.coef <- summary(ses.fit.model5a)
ses.fit.model5a.coef$model
```

```
## Simple exponential smoothing
##
## Call:
##  ses(y = trainL.ts, h = term, alpha = 0.01)
##
##   Smoothing parameters:
##     alpha = 0.01
##
##   Initial states:
##     l = 98.0139
##
##   sigma:  62.3491
##
##      AIC     AICc      BIC
## 1068.466 1068.614 1073.328
```

Based on the AIC and RMSE, Model5a is better than Model5b. Record model5a's performance as Model 5's

Models Performance Table

Method	AIC	RMSE
Model 1 - auto.arima ARIMA(0,0,0)	933.9	61.3
Model 2 - ARIMA(0,0,1)	935.7	61.3
Model 3 - ARIMA(0,0,0) with Fourier K=1	205.7	63.2
Model 4 - ARIMA(0,0,0) w/ Transformation	202.8	64.7
Model 5 - SES	1073.0	61.6

Notice how high AIC value is for model 5. It might not be a good idea to compare Model5's AIC with other models. Fitted model5 is based on the ses() function which uses means of data while other models whose coefficients have been estimating using maximum likelihood (ML).

It is also worthy to note that observations are lost with differencing or with lagging; therefore, we should not compare the AIC of an ARIMA model with differencing to one without differencing. (Hyndman 2013)

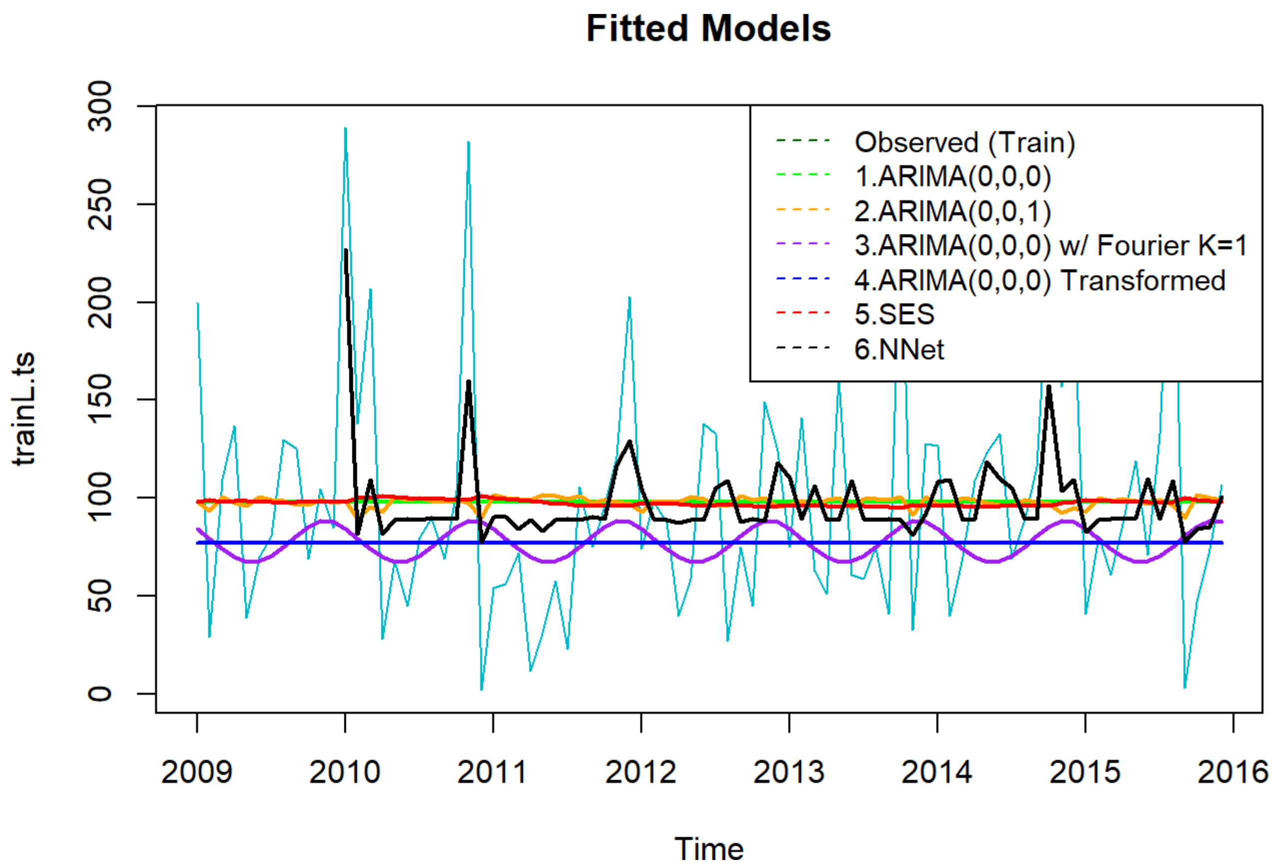
Model 6 - Neural Network Auto-Regressive

We will fit one more model, Model 6 - NNEtar: Neural Network Auto-Regressive Time Series Forecast.
NNetar is a feed-forward neural networks with a single hidden layer and lagged inputs for forecasting univariate time series.
Univariate is a term commonly used in statistics to describe a type of data which consists of observations on only a single characteristic or attribute. A simple example of univariate data would be the annual liver caner number. Neural networks work better at predictive analytics because of the hidden layers. Linear regression models use only input and output nodes to make predictions. The neural network also uses the hidden layer to make predictions more accurate.(Warudkar 2020)

```
nnetar.fit.Model6 <- nnetar(trainL.ts)
summary(nnetar.fit.Model6)
```

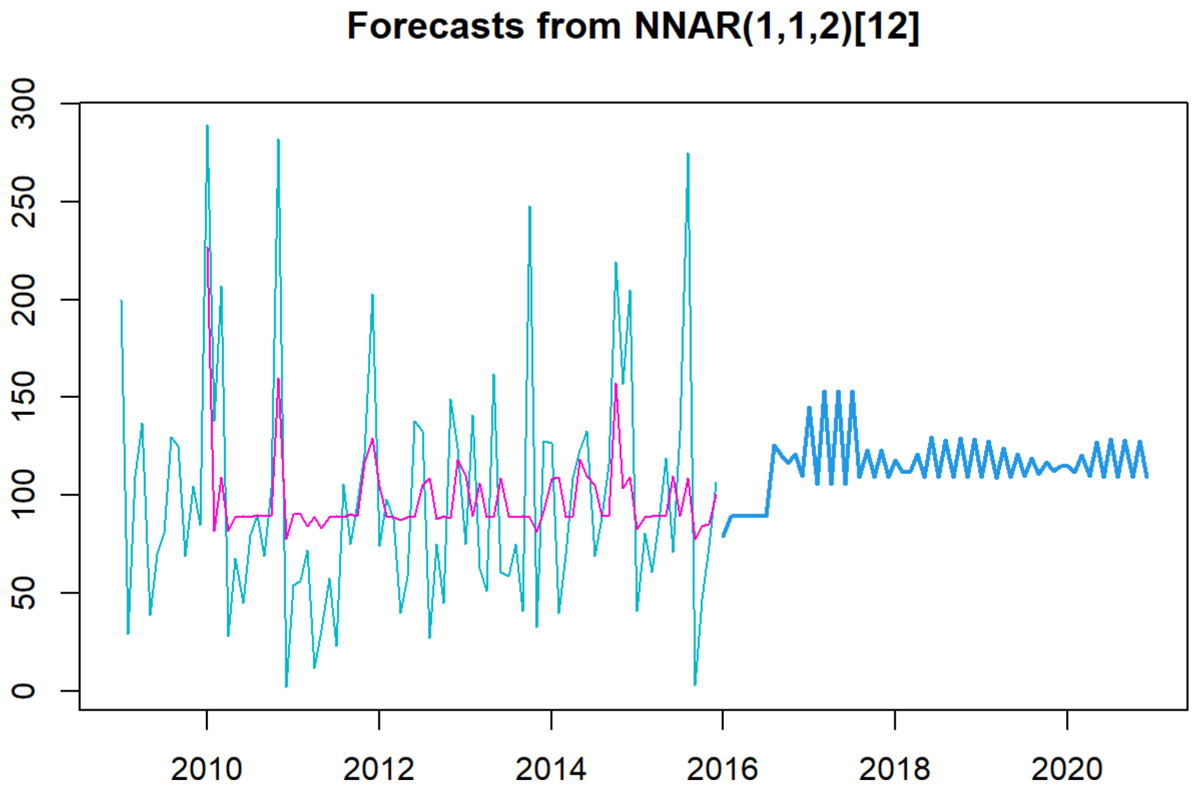
```
##           Length Class        Mode
## x         84     ts           numeric
## m          1     -none-       numeric
## p          1     -none-       numeric
## P          1     -none-       numeric
## scalex     2     -none-       list
## size       1     -none-       numeric
## subset    84     -none-       numeric
## model     20     nnetarmodels list
## nnetargs   0     -none-       list
## fitted    84     ts           numeric
## residuals 84     ts           numeric
## lags       2     -none-       numeric
## series     1     -none-       character
## method     1     -none-       character
## call       2     -none-       call
```

Plot fitted models



Plot the forecast

```
plot(forecast(nnetar.fit.Model6, h = term), col = "#00B7c7")
points(fitted(nnetar.fit.Model6), type = "l", col = "#FF00CC")
```



The prediction for model 6 seems to be more volatile than other models; however it also averages out to ~150 cases per month which is very close to the actual average cases.

```
nnetar.fit.Model6.fcast <- forecast(nnetar.fit.Model6, h = term)
round(nnetar.fit.Model6.fcast$mean, 1)
```

```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
## 2016  79.0  89.6  89.6  89.7  89.7  89.7  89.6 125.9 120.8 116.6 121.2 110.1
## 2017 145.0 105.9 153.2 105.8 153.2 105.8 153.2 109.5 123.2 109.4 123.3 109.5
## 2018 118.0 112.0 112.3 120.9 109.5 129.8 109.4 128.2 109.2 129.1 109.2 129.0
## 2019 109.1 128.0 109.1 124.4 109.4 120.9 110.1 119.1 110.9 117.2 112.3 114.8
## 2020 115.0 112.0 120.5 109.7 127.0 109.2 128.7 109.1 128.5 109.1 127.8 109.1
```

Model 1 and model 5 forecast are closely similar\ Model 1 averages out at 97.6, \ Model 5 at 98.3, and \ model 6 at ~150\ The higher the forecast the closer it is to the actual data.

Check Model 6 forecast accuracy

```
nnetar.fit.Model6.fcast.em <- nnetar.fit.Model6 %>%
  forecast(h = term) %>%
  accuracy(validL.ts)

round(nnetar.fit.Model6.fcast.em[, c("RMSE", "MAPE")], 1)
```

```
##              RMSE  MAPE
## Training set 52.6 146.8
## Test set     64.5 348.5
```

It seems a huge prediction difference between training and validation data set. Maybe there is an overfitting issue with this model. Record Model 6 performance.

Models Performance Table

Method	AIC	RMSE
Model 1 - auto.arima ARIMA(0,0,0)	933.9	61.3
Model 2 - ARIMA(0,0,1)	935.7	61.3
Model 3 - ARIMA(0,0,0) with Fourier K=1	205.7	63.2
Model 4 - ARIMA(0,0,0) w/ Transformation	202.8	64.7
Model 5 - SES	1073.0	61.6
Model 6 - nnetar	NA	52.6

Based on the RMSE, model 6 fairs very well compared to other models. We'll declaring Model 6 the best fitted model.

6. Validate the Best Fitted Model

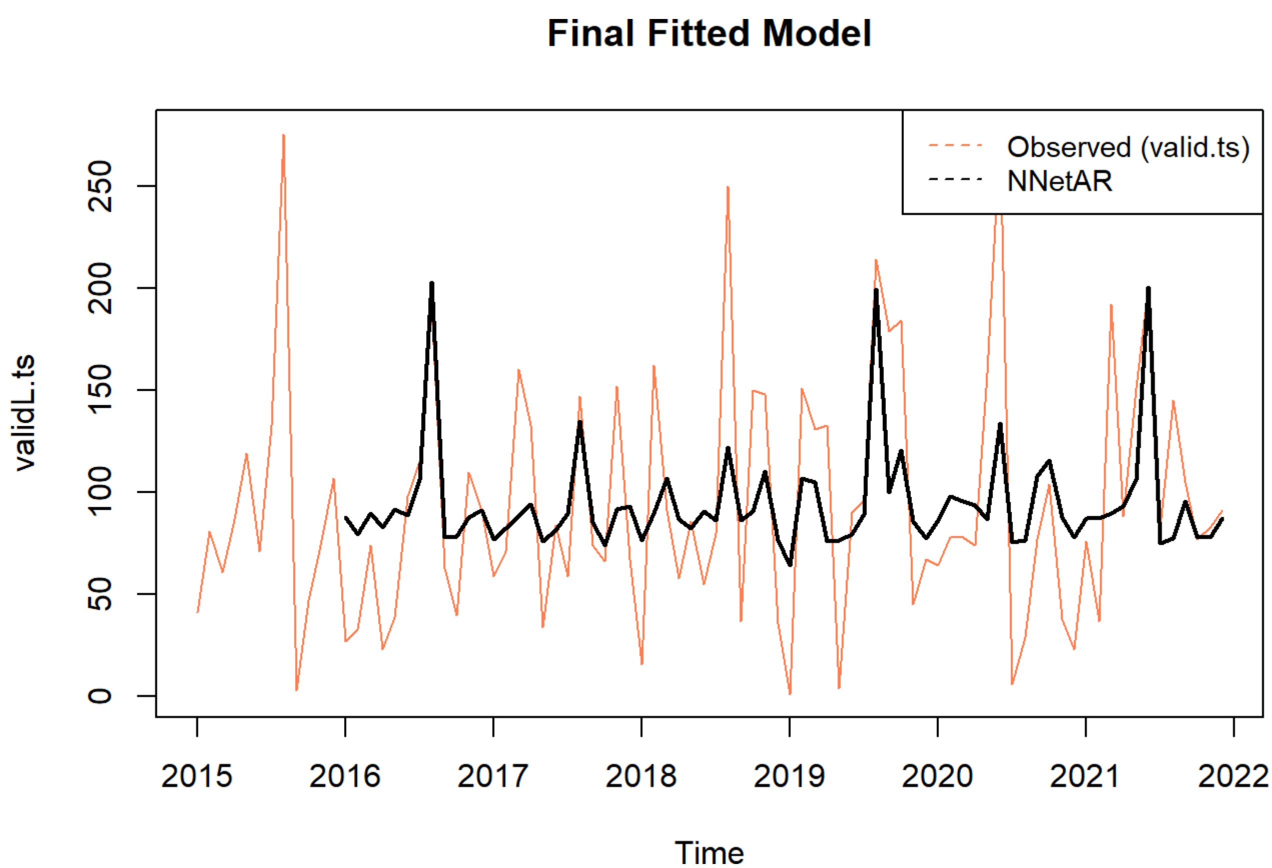
Validate Model 6 against the hold-out-set

```
nnetar.fit.model.final <- nnetar(validL.ts)
summary(nnetar.fit.model.final)
```

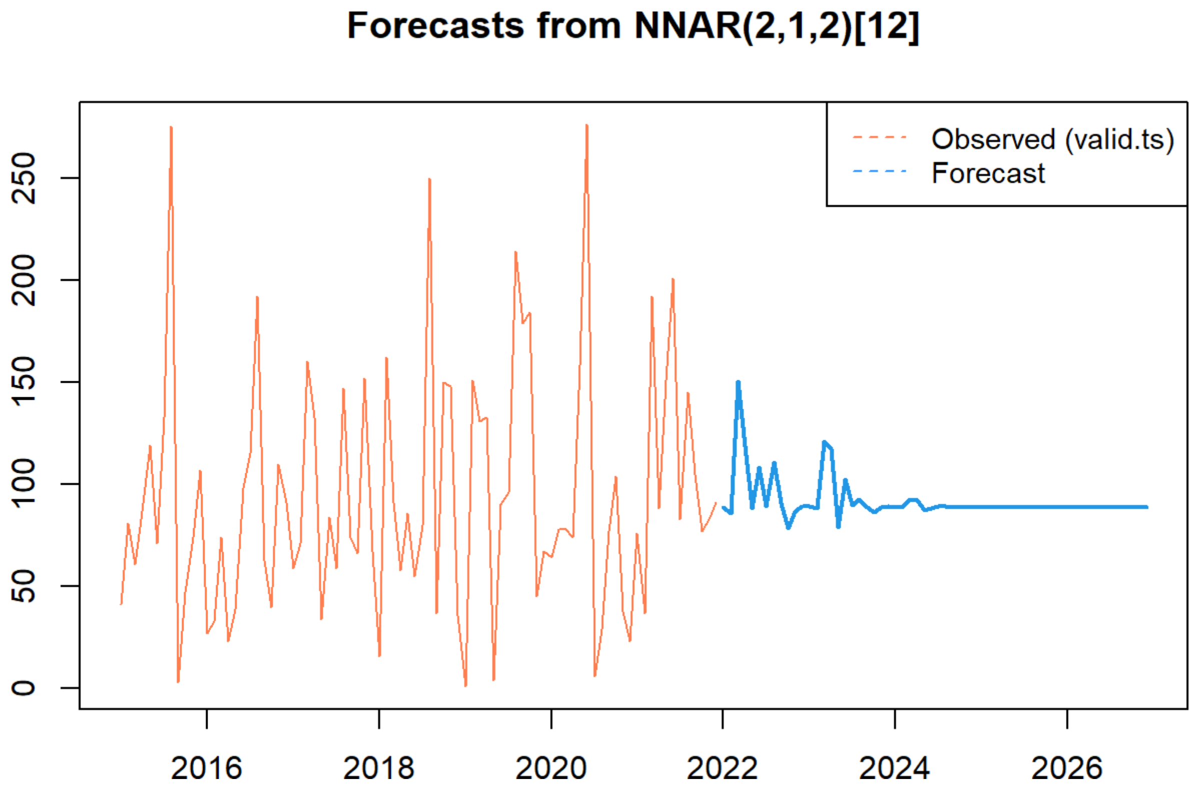
```
##            Length Class        Mode
## x          84     ts           numeric
## m           1     -none-       numeric
## p           1     -none-       numeric
## P           1     -none-       numeric
## scalex      2     -none-       list
## size        1     -none-       numeric
## subset     84     -none-       numeric
## model      20     nnetarmodels list
## nnetargs    0     -none-       list
## fitted     84     ts           numeric
## residuals  84     ts           numeric
## lags        3     -none-       numeric
## series      1     -none-       character
## method      1     -none-       character
## call        2     -none-       call
```

Plot the best fitted model

```
# Plot fitted models
plot(validL.ts, col = "#FF7f50", main = "Final Fitted Model")
lines(fitted(nnetar.fit.model.final), col = "black", lwd = 2)
legend("topright", c("Observed (valid.ts)", "NNetAR"), lty = 8, col = c("#FF7F50", "black"), c
ex = 0.9)
```




```
plot(forecast(nnetar.fit.model.final, h = term), col = "#FF7F50")
legend("topright", c("Observed (valid.ts)", "Forecast"), lty = 8, col = c("#FF7F50", "#3399FF"
), cex = 0.9)
```




```
nnetar.fit.model.final.fcast <- forecast(nnetar.fit.model.final, h = term)
round(nnetar.fit.model.final.fcast$mean, 1)
```

```
##          Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
## 2022    88.7  85.8 150.4 118.3  88.2 108.1  89.2 110.9  90.8  78.5  87.0  89.3
## 2023    89.0  88.4 121.0 117.3  78.8 102.5  89.7  92.8  88.8  86.5  88.5  89.0
## 2024    88.8  88.6  92.4  92.3  87.1  88.0  89.0  89.0  88.7  88.5  88.7  88.8
## 2025    88.7  88.7  88.9  88.9  88.6  88.6  88.7  88.7  88.7  88.7  88.7  88.7
## 2026    88.7  88.7  88.7  88.7  88.7  88.7  88.7  88.7  88.7  88.7  88.7  88.7
```

The neural networks (Nnetar) time series forecasts show a monthly flux trend in the number of liver cancer cases. On a monthly average the maximum and minimum are:

```
## [1] 150.4
```

```
## [1] 78.5
```

Average max and min cases per year

Overview of the forecast values

fy	avg.max	pct.max.change	avg.min	pct.min.change
2022	150.4	NA	78.5	NA
2023	121.0	-19.5	78.8	0.4
2025	88.9	-3.8	88.6	1.7
2026	88.7	-0.2	88.7	0.1

The Overview of the Forecast Values table shows the fy, average max, percent max change, average min, and percent min change of the liver cancer cases.

```
## RMSE  MAPE
##  46.3 180.2
```

The assessment tells us that on an average month, the predictions are off by 4.5 liver cases or around 16%. Our scale is set in thousands.

Record the final model performance and compare.

Models Performance Table

Method	AIC	RMSE
Model 1 - auto.arima ARIMA(0,0,0)	933.9	61.3
Model 2 - ARIMA(0,0,1)	935.7	61.3
Model 3 - ARIMA(0,0,0) with Fourier K=1	205.7	63.2
Model 4 - ARIMA(0,0,0) w/ Transformation	202.8	64.7
Model 5 - SES	1073.0	61.6
Model 6 - nnetar	NA	52.6
Model6 Final - nnetar	NA	46.3

7. Conclusion

Actual Cases 2009-2021 —vs– Forecast Cases 2022-2026

fy	case_count	avg_count	percent_change
2009	5267	92.4	NA
2010	5524	95.2	4.9
2011	6026	103.9	9.1
2012	6809	113.5	13.0
2013	7223	118.4	6.1
2014	7948	134.7	10.0
2015	9124	147.2	14.8
2016	9023	147.9	-1.1
2017	9443	154.8	4.7
2018	9821	158.4	4.0
2019	9549	151.6	-2.8
2020	9908	154.8	3.8
2021	9297	145.3	-6.2

fy	avg.max	pct.max.change	avg.min	pct.min.change
2022	150.4	NA	78.5	NA
2023	121.0	-19.5	78.8	0.4
2024	92.4	-23.6	87.1	10.5
2025	88.9	-3.8	88.6	1.7
2026	88.7	-0.2	88.7	0.1

Selecting Model 6 NNetar as our final may be a good choice. It is accurately characterized the trend of liver cancer volume.

Examine the Actual Cases 2009-2021 and Forecast Cases 2022-2026 above side by side. Based on the actual **avg_count cases and the percent change** with the forecast **avg.max cases and percent change**, our model prediction is pretty accurate.

On average, from 2009 to 2021 the liver cancer cases gradually increased from 92 to 145 cases; from 2021 to 2022 the cases increase to 159, and from 2023 to 2026, it gradually decreases from 136 to 101 cases.

Fitting ARIMA model is more of an art than a science (weecology 2021). In reality, over two dozen models were fitted but only six are presented in this project.

Additional Work

- We acknowledge that the time frame of this project is a limitation. Specifically, liver cancer data were only available from 2009 to 2021.
- Include Box-Cox Transformation in the model fitting process. Box-Cox method helps to address non-normally distributed data by transforming to normalize the data. When the assumption of data normally distributed is violated or the relationship between the dependent and independent variables in case of linear model are not linear, In such situations some transformations methods that may help the data set follow a normal distribution. It's worthy to note that a value of $\lambda=0$ corresponds to the multiplicative decomposition while $\lambda=1$ is equivalent to an additive decomposition. You can use a Box-Cox Transformation by setting lambda = 0 because the variance increases with the level of the series.(Dynamic harmonic regression by datacamp).
- Clean any outliers using tsclean(), if necessary impute any missing values. Time Series data have a continuity and a dependence and having any missing values will affect your model severely.
- Additional data and trend analysis would be helpful including lag.plot,
- Perform decompose() to isolate irregular data and seasonal, if there are seasonal signals in the data.
- Future work could examine how the time trends could change according to specific demographic subgroups and geographic regions.

Lesson Learned

- The AIC penalizes complex models. A certain penalty for complex models is necessary to avoid overfitting of our statistical models. Overfitting is an undesirable machine learning behavior that occurs when the model gives accurate predictions for training data but not for new data or hold-out-set data. To prevent model overfitting, it's a good idea to train the model on a known data set before making prediction.
- When fitting model ARIMA(1,0,22), we discovered that it's almost identical to Model 2 ARIMA(0,0,22) based on the AIC.
- It may not be a good idea to include Fourier terms if there are not any seasonality in the data. For long term forecasting **seasonality** has to take into account as well as using smoothness and regressing on a few Fourier terms. See illustration by (Scortchi-Reinstate Monica, 2017)
- The output of your models is only as good as your input. Adding regressors to an ARIMA model only makes sense if there is some clear correlation between the variables. The auto.arima() function handles regression terms via the xreg argument.
- Arima() will fit a regression model with ARIMA errors if the argument xreg is used. The order argument specifies the order of the ARIMA error model. If differencing is specified, then the differencing is applied to all variables in the regression model before the model is estimated. (Hyndman,9.2)
- Relative model performance metrics

  a. Akaike Information Criterion (AIC), shows you how good a model is relative to the other models. AIC penalizes complex models (with more parameters) in favor of simple ones.

     - AIC calculated formula: $AIC = 2k - 2Ln (\hat{L})$ Where k is the number of parameters in the model, L-hat is the maximum value of the likelihood function for the model, and ln is the natural logarithm.

  b. Bayesian Information Criterion (BIC) is an estimate of a function of the posterior probability of a model being true under a certain Bayesian setup. Once again, the lower the value, the better the model.

     - BIC calculated formula: BIC = kln(n) - 2Ln$(\hat{L})$ Where k is the number of parameters in the model, $\hat{L}$ is the maximum value of the likelihood function for the model, n is the number of data points (sample size), and ln is the natural logarithm.

  both AIC and BIC are relative metrics, so you can't directly compare models for different datasets. Instead, choose the model with the lowest score.

- General regression metrics

  a. RMSE — Root Mean Squared Error

     - RMSE tells you how many units your model is wrong on average. In our airline passengers example, the RMSE will tell you how many passengers you can expect the model to miss in every forecast.

  b. MAPE — Mean Absolute Percentage Error

     - MAPE tells you how wrong your forecasts are percentage-wise. I like it because, in a way, it is equivalent to accuracy metric in classification problems. For example, the MAPE value of 0.02 means your forecasts are 98% accurate. (Dario 2021)

References

Boateng, Nana, Time Series Analysis Methods, Building Arima Models for Forecasting, https://rstudio-pubs-static.s3.amazonaws.com/303786_f1b99d6b7e9346c4b1488a174bab839a.html.

Brown, Kyle W., Nov 25, 2020, Download CSV from Github with R, https://rpubs.com/kylewbrown/github-csv-r.

Burns Emily, Jan 14, 2021, Data Cleaning in R Made Simple, https://towardsdatascience.com/data-cleaning-in-r-made-simple-1b77303b0b17.

Chen, James, March 18, 2022, What is EMA? How to use Exponential Moving Average with Formula, https://www.investopedia.com/terms/e/ema.asp.

Cheryl, Jan 21, 2020, Handling Skewed data for Machine Learning models, https://reinec.medium.com/my-notes-handling-skewed-data-5984de303725.

cylurian, 2011, *Skewed Histogram (Left Skewed Right Skewed Histogram)*, https://www.youtube.com/watch?v=H9ITfdaX2ZQ.

DataCapmp, *Dynamic harmonic regression*, https://campus.datacamp.com/courses/forecasting-in-r/advanced-methods?ex=4.

DataCamp, fourier: Fourier terms for modelling seasonality, https://www.rdocumentation.org/packages/forecast/versions/8.12/topics/fourier.

Geeksforgeeks, Jun 20, 2022, Exponential Smoothing in R Programming, https://www.geeksforgeeks.org/exponential-smoothing-in-r-programming/.

Hyndman, Rob J. and Athanasopoulos, George, Forecasting: Principles and Practice (2nd ed), 8.1 Stationarity and differencing, https://otexts.com/fpp2/stationarity.html.

Hyndman, Rob J., Principles and Practice (2nd ed), 9.2 Regression with ARIMA errors in R, https://otexts.com/fpp2/regarima.html.

Hyndman, Athanasopoulos, Bergmeir et al, Neural Network Time Series Forecasts, https://pkg.robjhyndman.com/forecast/reference/nnetar.html#:~:text=Feed%2Dforward%20neural%20networks%20with,for%20forecasting%20univariate%20time%20series.

Hyndman, Rob J., Jul 4, 2013, Facts and fallacies of the AIC, https://robjhyndman.com/hyndsight/aic/.

International Institute of Forecasters, international journal of forecasting(ijf), https://forecasters.org/ijf/awards/.

Stack Exchange, June 20, 2016, Jeremias K., What selection critieria to use and why? (AIC, RMSE, MAPE) - All possible model selection for time series forecasting, https://stats.stackexchange.com/questions/219747/what-selection-criteria-to-use-and-why-aic-rmse-mape-all-possible-model-s#:~:text=They%20have%20the%20drawback%20of,a%20better%20value%20for%20RMSE.

Khikmah, Khusnia Nurul, Jan 31, 2022, Simple Moving Average, Task 2 Time Series Analysis Practicum,
https://rpubs.com/khusniank/SMAADWprac2.

Losada, Luis, Feb 3, 2020, Time Series Analysis with Auto.Arima in R,
Regression with ARIMA Errors using Auto Arima,
https://towardsdatascience.com/time-series-analysis-with-auto-arima-in-r-2b220b20e8ab.

Ludlow, Jorge & Enders Walter, September 2000, International Journal of Forecasting, Estimating non-linear ARMA
models using Fourier coefficients,https://www.sciencedirect.com/science/article/abs/pii/S0169207000000480.

Mithcell, Cory, Nov 17, 2021, Displace Moving Average (DMA): What it is, How Traders Use it,
https://www.investopedia.com/terms/d/displacedmovingaverage.asp.

National Cancer Institute (NCI), Sept 1, 2021, Cancer Health Disparities Research,
https://www.cancer.gov/research/areas/disparities.

The Office of Cancer Centers (OCC), NCI, 2023, Data Table 3, https://cancercenters.cancer.gov/DT/DT3.

NIST, April, 2012, Engineering Statistics Handbook,
https://www.itl.nist.gov/div898/handbook/eda/section3/eda33f.htm.

National Library of Medicine, National Center for Biotechnology Information (NCBI), Aug 19, 2019, Forecasting annual incidence and mortality rate for prostate cancer in Australia until 2022 using autoregressive integrated moving average (ARIMA) models,
https://www.ncbi.nlm.nih.gov/pmc/arti cles/PMC6707661/.

Radecic, Dario, Jul 26, 2021, Time Series From Scratch — Train/Test Splits and Evaluation Metrics,
https://towardsdatascience.com/time-series-from-scratch-train-test-splits-and-evaluation-metrics-4fd654de1b37.

Rasheed, Rayhann, July 11, 2020, Why Does Stationarity Matter in Time Series Analysis?,
https://towardsdatascience.com/why-does-stationarity-matter-in-time-series-analysis-e2fb7be74454.

SelcukDisci, Oct 24, 2020, Approaches to Time Series Data with Weak Seasonality,
https://www.datasciencecentral.com/approaches-to-time-series-data-with-weak-seasonality/.

Singh, Deepika, July 12, 2019, Time Series Forecasting Using R,
https://www.pluralsight.com/guides/time-series-forecasting-using-r.

Stack Exchange, June 10, 2019, How to read checkresiduals graphics in R?
https://stats.stackexchange.com/questions/386875/how-to-read-checkresiduals-graphics-in-r.

Stack Exchange, 2017, Scortchi-Reinstate Monica, Seasonality not taken account of in auto.arima(),
https://stats.stackexchange.com/questions/213201/seasonality-not-taken-account-of-in-auto-arim

a).

Tech Know How, 2020, *Stock Price Forecasting - Part 1- Loading the Data and Preparing it!*, https://www.youtube.com/watch?v=P-I0ljQpRCI.

Tracyrenee, Aug 1, 2022, AIC vs RMSE When Gridsearching SARIMAX Hyperparemeters, https://python.plainenglish.io/aic-vs-rmsc-when-gridsearching-sarimax-hyperparemeters-e1c8d5c40578.

Warudkar, Hemant, Feb 27, 2020, Prediction using Neural Networks, https://www.expressanalytics.com/blog/neural-networks-prediction/.

weecology, 2021,*Introduction to Time Series Decomposition*,https://www.youtube.com/watch?v=D45ddUVcEfY.

weecology, 2021, *Modeling seasonal signals in ARIMA models*, https://www.youtube.com/watch?v=m6fpl0po4qs.

weecology, 2021, *Using auto.arima() in r*, https://www.youtube.com/watch?v=JUIoMc0isdI

zhou, Nai Biao, March 11, 2021, Exploring Four Simple Time Series Forecasting Methods with R Examples, https://www.mssqltips.com/sqlservertip/6778/time-series-forecasting-methods-with-r-examples/.

Zach, May 25, 2021, Augmented Dickey-Fuller Test in R (With Example), https://www.statology.org/dickey-fuller-test-in-r/.