Lab 4

Ram Balasubramanian, John Kenney

November 28, 2017

Read and Examine the data:

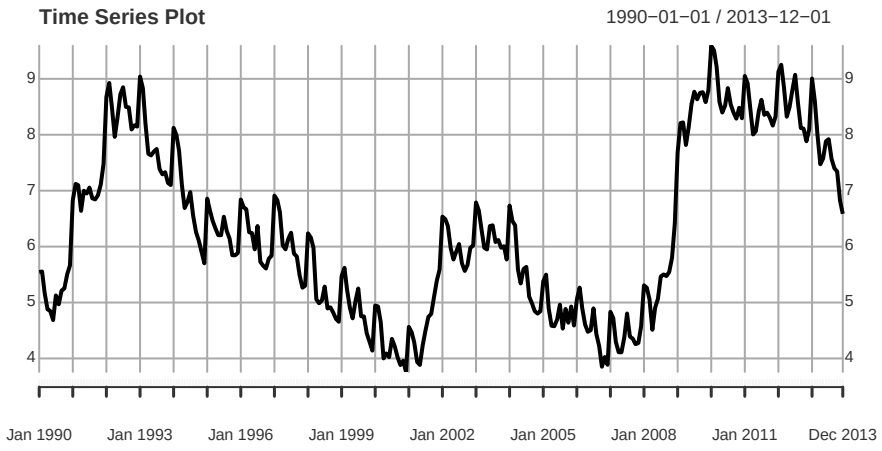
We conducted a thorough examination of the data to ascertain there aren't any missing values, coding errors etc. We did not identify any issues that needed to be addressed. We have not provided the details of that examination to reserve the allotted page-limit for more interesting examination of the given time series.

```
x = as.data.frame(read.csv("Lab4-series2.csv", header = TRUE))
colnames(x) <- c("num", "value")
idx <- seq(as.Date("1990/1/1"), by = "month", length.out = dim(x)[1])
# Split Timeseries into Train,Development, and Test sets
ts_all <- xts(x["value"], order.by = idx, frequency = 12)
ts.train = ts_all["1990-01-01/2013-12-31"]  #Create train set from 1990-2013
ts.dev = ts_all["2014-01-01/2014-12-31"]  #we will use a dev set to tune hyper parameters
ts.test = ts_all["2015-01-01/"]  #we will use test set to measure model performance
```

1. Exploratory Data Analysis:

1.1 Plot the time series

```
plot.xts(ts.train, type = "l", minor.ticks = NULL, major.ticks = 12, main = "Time Series Plot",
    grid.ticks.on = "years")
```



Observations:

Trend: Plot doesn't show any obvious signs of a trend. Series exhibits patterns where there are sudden shocks, when the series goes up quickly; and reverts back very slowly.
Seasonal/Cyclical patterns: There appear to be seasonal ups and downs (which are very apparent especially

during years when the series is relatively stable e.g. '95-'98).
Outliers: There aren't any apparent outliers.
Transformations: We do not see any patterns in the time-series that suggests that we should transform the data (like a log, or power transformation). The series variance seems to be stable over time.

Next step in our EDA, we will check for stationarity.

1.2 Check for Stationarity:

1.2.1 Stationarity - Is there a trend?:

Though visual inspection of the time series plot showed no trend, we employed smoothing techniqes to plot a smoothed version of the series (which did not show any trend as well). We also fit a linear model to see if there is a statistically significant non-zero slope coefficient.

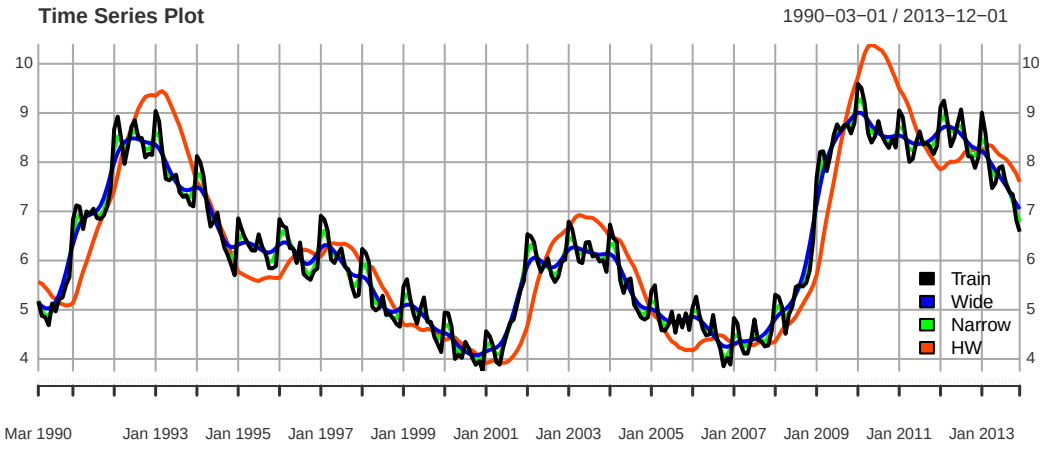
```
# Add Smoothing
library(fpp)
```

```
## Loading required package: fma
```

```
## Loading required package: expsmooth
```

```
## Loading required package: lmtest
```

```
N = length(ts.train)
k.smooth.wide = ksmooth(time(ts.train), ts.train, kernel = c("normal"),
    bandwidth = 200)
ksw = as.xts(k.smooth.wide$y, order.by = index(ts.train))
k.smooth.narrow = ksmooth(time(ts.train), ts.train, kernel = c("normal"),
    bandwidth = 100)
ksn = as.xts(k.smooth.narrow$y, order.by = index(ts.train))
hw = HoltWinters(ts.train, alpha = 0.1, beta = 0.2, gamma = F)
hw.xhat = hw$fitted[, "xhat"]
hw.xts = xts(hw.xhat, order.by = index(ts.train[3:N]))
tw = cbind(ts.train[3:N], ksw[3:N], ksn[3:N], hw.xts)
colnames(tw) = c("Train", "Wide", "Narrow", "HW")

plot.xts(tw, type = "l", col = c("black", "blue", "green", "orangered"),
    minor.ticks = NULL, major.ticks = 12, main = "Time Series Plot", grid.ticks.on = "years",
    xlim = c(min(index(ts.train)), max(index(ts.train))), legend.loc = "bottomright")
```



```
trnd = lm(x, formula = value ~ num)
coef(summary(trnd))
```

```
##             Estimate Std. Error t value   Pr(>|t|)
## (Intercept) 5.962719   0.168476  35.392 9.909e-111
## num         0.001993   0.000936   2.129  3.402e-02
```

Observations:

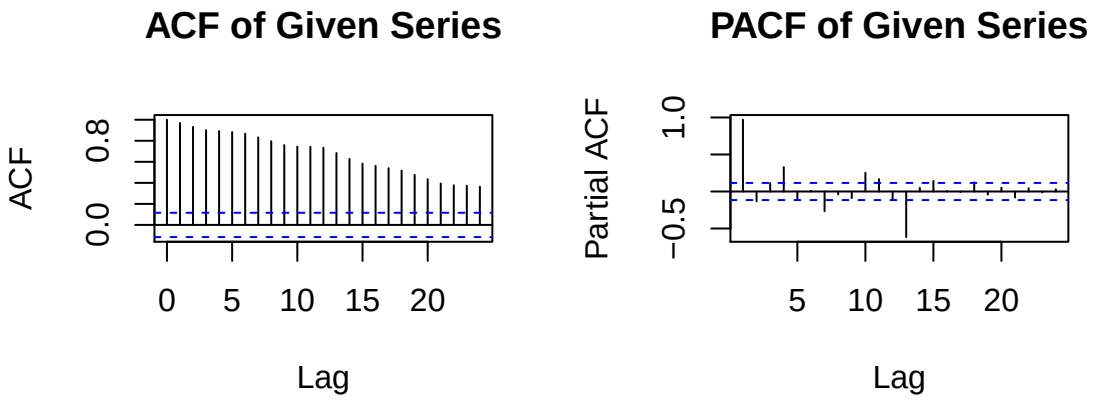
Trend: Series doesn't show any overall trend; We **do** see a statistically significant linear trend - but the slope coefficient is very small and is **practically insignificant**. So we say the series is **Trend Stationary**.

1.2.2 Stationarity: Is there Unit Root present?:

```
# conduct Dicky-Fuller test to see if series is unit root stationary
adfPvalue = adf.test(ts.train)$p.value
result = "Unit Root Present - Non Stationary Series"
if (adfPvalue < 0.05) {
    result = " Reject Unit Root Presence - Stationary Series"
}
print(paste(result, " - p-value of adf-test =", round(adfPvalue, 4)))
```

```
## [1] "Unit Root Present - Non Stationary Series  - p-value of adf-test = 0.5433"
```

```
# Look at the acf and pacf plots
par(mfrow = c(1, 2))
acf(ts.train, main = "ACF of Given Series")
pacf(ts.train, main = "PACF of Given Series")
```

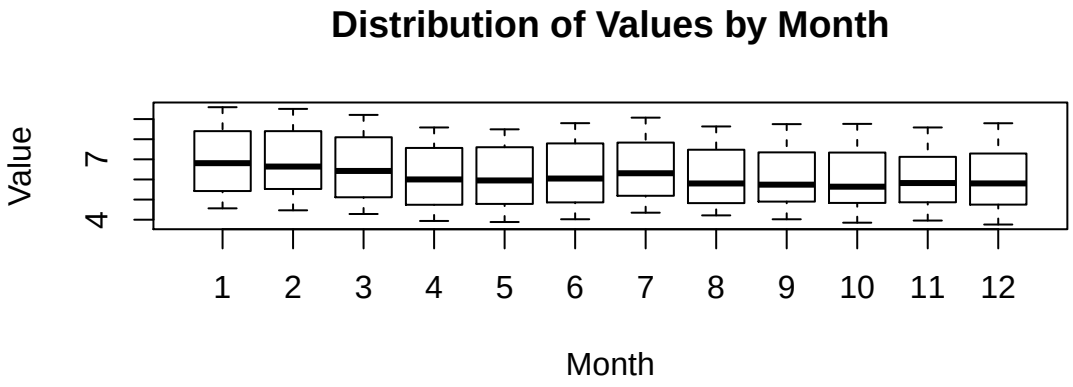


Observations:

Based on the p-value for the Dicky-Fuller test, we **cannot reject the Null Hypothesis: "Unit root is present"** . The ACF declines very very slowly and The PACF shows significance at lag-1. This also suggests Random Walk behavior. So we should consider a first difference in order to make the series stationary.
Other observations - there is also significant correlations at lags 4,7,10,11,13 and the correlation at lags 7 and 13 is negative.

1.2.3 Stationarity: Do we see Seasonality in the data?:

Seasonal decomposition using decompose() and stl() show no evidence of seasonality, but visual inspection of the series definitely shows seasonal patterns. Let's confirm that with a plot of the series by months, and computing the monthly means and doing a statistical test to see if the means are the same.




```
##                          Df Sum Sq Mean Sq F value Pr(>F)
## month(index(ts.train))    1     16   16.23    7.08 0.0083 **
## Residuals               286    656    2.29
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Observations:

The boxplot shows the values of the time series for each month fluctuates around a different mean each month. This suggests that the series has seasonal patterns. The Anova test says we can reject the Null Hypothesis that the means by month are equal. There is clearly Seasonality in the series that needs to be addressed.

2. Modeling:

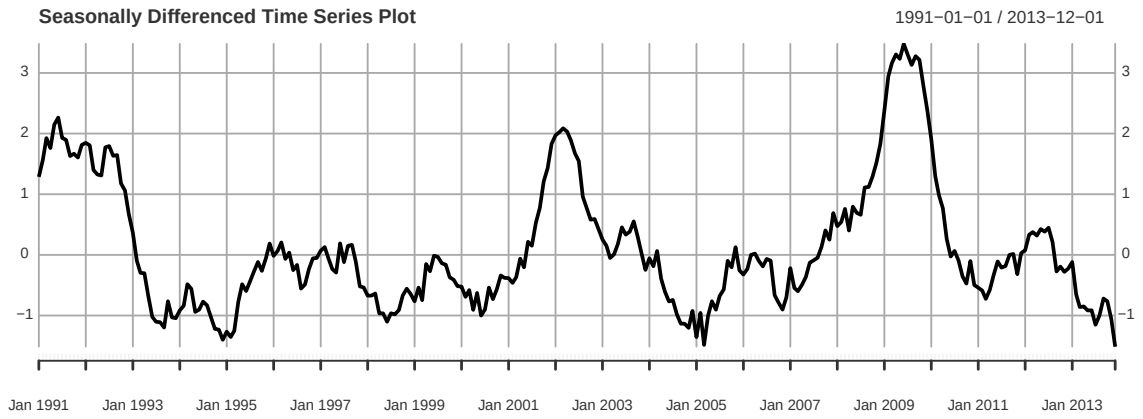
2.1 Addressing Non-Stationarity:

We have identified two types of potential stationarity - Seasonality and Random-Walk. Let's address Seasonality first.

2.1.1 Seasonal Stationarity:

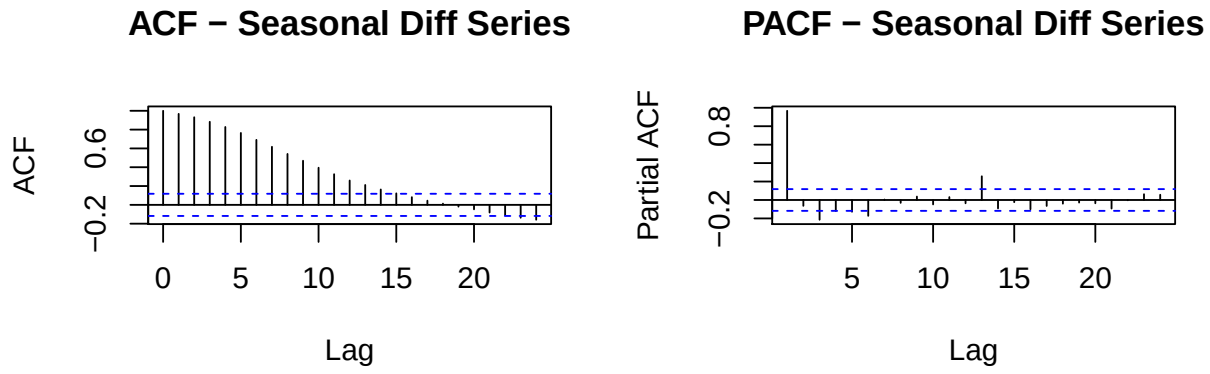
Given the pattern observed in the series, a 12-m difference seems appropriate.

```
diff12 = diff(ts.train, lag = 12, differences = 1)["1991-01-01/"]
plot.xts(diff12, type = "l", minor.ticks = NULL, major.ticks = 12, main = "Seasonally Differenced Time S
    grid.ticks.on = "years")
```



Let's study the behavior of this differenced series - is it stationary now?

```
par(mfrow = c(1, 2))
acf(diff12, na.action = na.pass, main = "ACF - Seasonal Diff Series")
pacf(diff12, na.action = na.pass, main = "PACF - Seasonal Diff Series")
```




```
adfPvalue = adf.test(diff12)$p.value
result = "Unit Root Present - Non Stationary Series"
if (adfPvalue < 0.05) {
    result = " Reject Unit Root Presence - Stationary Series"
}
print(paste(result, " - p-value of adf-test =", round(adfPvalue, 4)))
```

```
## [1] " Reject Unit Root Presence - Stationary Series  - p-value of adf-test = 0.0214"
```

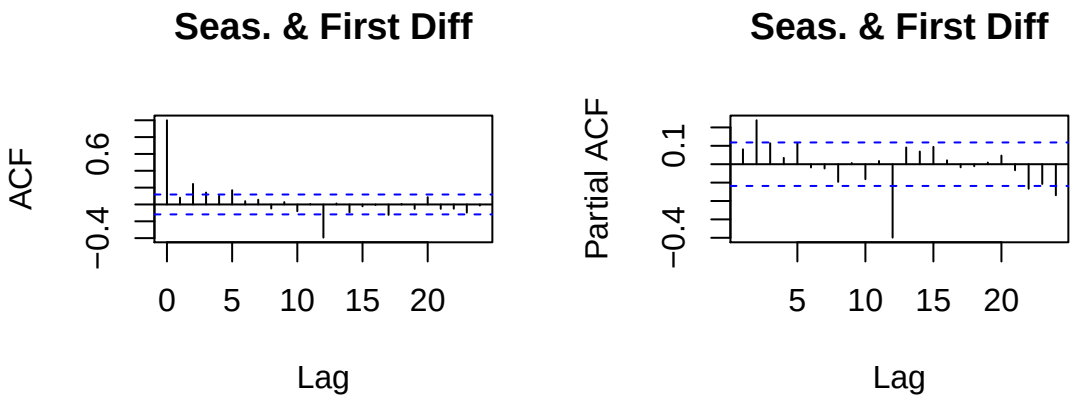
Observations:

The acf shows a pattern of tailing off and the pacf plot shows very high correlation at lag-1 and a significant negtive correlation at lag 3, 6; and positive at 13. This suggests that we may have an AR process; but the ACF tails of fairly slowly suggesting that we might benefit from a first difference.

The ADF test shows that there is no unit-root present. So based on the examination of the series after seasonal differencing - **we can either difference it again, or go with an AR model of a higher order**. Let's study next the effect of first differencing on the Seasonally Differenced series.

2.1.2 First Difference of Seasonally Differenced Series

```
diff1 = diff(diff12, lag = 1, differences = 1, na.pad = F)
par(mfrow = c(1, 2))
acf(diff1, na.action = na.pass, main = "Seas. & First Diff")
pacf(diff1, na.action = na.pass, main = "Seas. & First Diff")
```




```
adfPvalue = adf.test(diff1)$p.value
result = "Unit Root Present - Non Stationary Series"
if (adfPvalue < 0.05) {
    result = " Reject Unit Root Presence - Stationary Series"
}
print(paste(result, " - p-value of adf-test =", round(adfPvalue, 4)))
```

```
## [1] " Reject Unit Root Presence - Stationary Series  - p-value of adf-test = 0.01"
```

Observations:

The plot of this differenced is more stationary, so likely does not require any further differencing. We see that there are some significant correlations in the first few lags of the ACF and a spike in the PACF at lag 2. **This suggests an AR[2] process**. The PACF also decreases from lag 2 to 7, **so there might be an underlying MA 1 term.** The PACF is negative at the seasonal period (12) and to a lesser extent (24), **which might indicate an order 1 seasonal MA term**. The ACF shows is a significant negative autocorrelation lag at (12) and possibly at lag (24), **which might indicate an order 1 seasonal AR term**

2.2 Model Parameter Search:

Given these observations, we will fit an ARIMA(p,d,q)x(P,D,Q)[12] model. We know for sure the following parameters: D = 1; d = 1. We also have evidence to try q = 1, Q = 1, p = 2, and P = 1. Our search space will be Arima(p=1,2 d=1 q=0,1)x(P=0,1 D=1 Q=0,1), 16 models total.

2.3: Model Selection Approach:

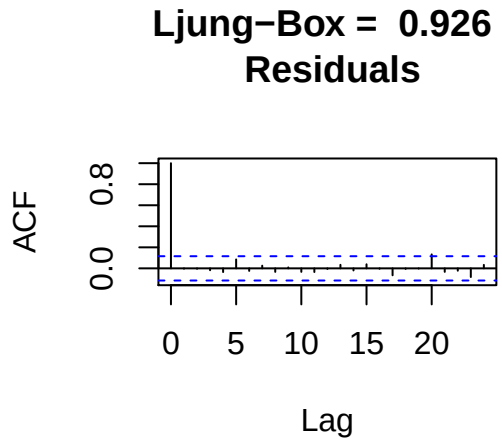
We will fit each of these models on the train set and evaluate performance using AIC. The four best AICs will then undergo residuals analysis using ACF/PACF and calculation of the Llung-Box p-value up to 24 lags, per Rob Hyndman for seasonal data (https://robjhyndman.com/hyndsight/ljung-box-test/).

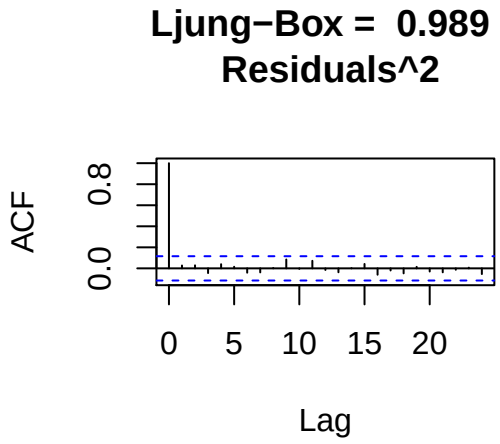
Following residuals analysis, we will calculate forecast accuracy on the 2014 development set using Symmetric MAPE, which addresses traditional MAPE's heavier penalties on negative predictions than positive ones (see https://robjhyndman.com/hyndsight/smape/). Finally, once we decide on the final model parameters, we will run the model through the test set to evaluate performance.

```
arima.loop = function(plist, dlist, qlist, Plist, Dlist, Qlist, Mlist) {
    df = data.frame(matrix(vector(), nrow = 0, ncol = 8, dimnames = list(c(),
        c("p", "d", "q", "P", "D", "Q", "m", "AIC"))), stringsAsFactors = T)
    for (m in Mlist) {
        for (Q in Qlist) {
            for (q in qlist) {
                for (D in Dlist) {
                  for (d in dlist) {
                    for (P in Plist) {
                      for (p in plist) {
                        tryCatch(expr = {
                          modfit = arima(ts.train, order = c(p, d, q),
                            seasonal = list(order = c(P, D, Q), period = m),
                            method = "ML")
                          mf.aic = modfit$aic
                          df[dim(df)[1] + 1, ] = c(p, d, q, P, D, Q, m,
                            mf.aic)
                        }, error = function(e) {
                          df[dim(df)[1] + 1, ] = c(p, d, q, P, D, Q, m,
                            NA)
                        })
                      }
                    }
                  }
                }
            }
        }
    }
    return(df)
}

# search for model in the parameter search space
results.df = arima.loop(plist = c(1, 2), dlist = c(1), qlist = c(0, 1),
    Plist = c(0, 1), Dlist = c(1), Qlist = c(0, 1), Mlist = c(12))

# Limit to top 4 Best AICs and analyze residuals
best.AIC = head(results.df[order(as.numeric(results.df$AIC)), ], 4)
best.AIC$Box.p = rep(0, length(best.AIC[, 1]))
best.AIC$Box.p.sq = rep(0, length(best.AIC[, 1]))
for (i in seq(length(best.AIC[, 1]))) {
    modfit = arima(ts.train, order = as.vector(as.numeric(best.AIC[i, 1:3])),
        seasonal = list(order = as.vector(as.numeric(best.AIC[i, 4:6])),
            period = as.numeric(best.AIC[i, 7])), method = "ML")
    best.AIC[i, 9] = round(Box.test(modfit$residuals, type = "Ljung-Box",
        lag = 24)$p.value, 3)
    best.AIC[i, 10] = round(Box.test(modfit$residuals^2, type = "Ljung-Box",
        lag = 24)$p.value, 3)
    print(best.AIC[i, 1:7], collapse = " ")
    par(mfrow = c(1, 2))
```

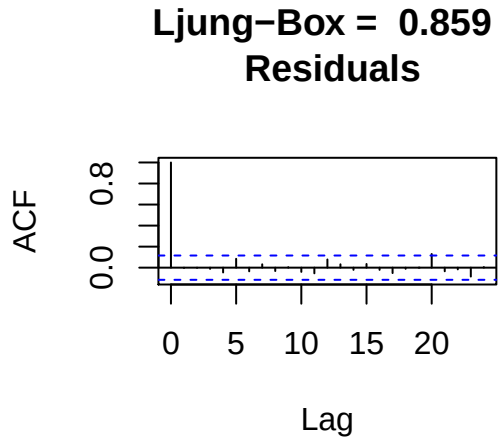
```
    acf(modfit$residuals, main = paste("Ljung-Box = ", best.AIC[i, 9],
        "\n Residuals"))
    acf(modfit$residuals^2, main = paste("Ljung-Box = ", best.AIC[i, 10],
        "\n Residuals^2"))
}
```

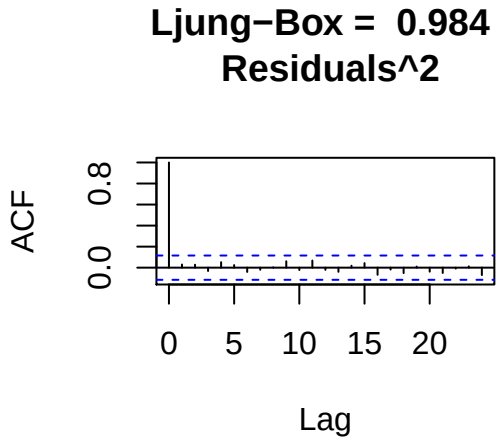
```
##    p d q P D Q  m
## 16 2 1 1 1 1 1 12
```



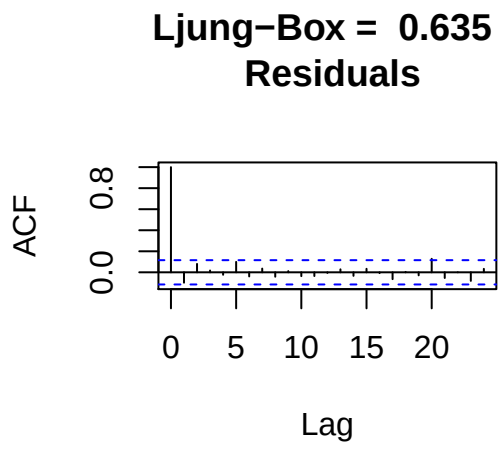


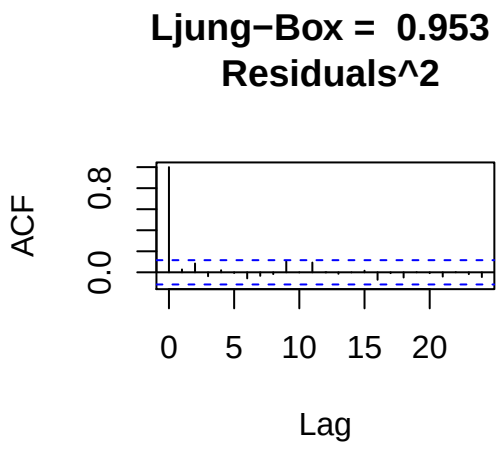

```
##    p d q P D Q  m
## 14 2 1 1 0 1 1 12
```



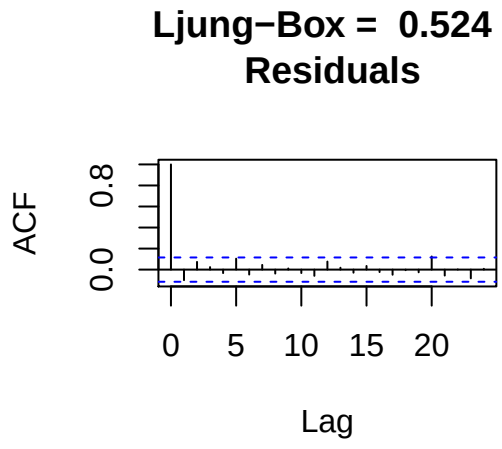


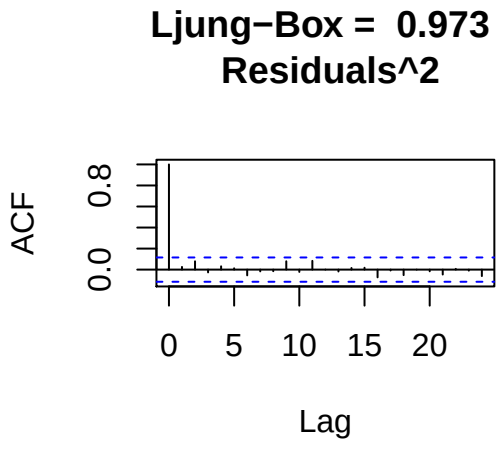

```
##    p d q P D Q  m
## 15 1 1 1 1 1 1 12
```






```
##    p d q P D Q  m
## 13 1 1 1 0 1 1 12
```





We see that the residuals for the top 4 AIC models are well-behaved. There isn't conclusive evidence that any residuals are auto-correlated. Furthermore, because the squared-residual series also show no auto-correlation, we can say that no series exhibits conditional-heteroskedasticity (Cowpertwait, pg. 146). Both of these statements are confirmed with ACF plots, which show no conclusively auto-correlated lags. Let's now look at how each of these models performs on the 2014 dev data set, using Symmetric MAPE to assess goodness-of-fit.

```
sMAPE = function(y.meas, y.pred) {
    s = c()
    for (i in seq(1, length(y.meas))) {
        s[i] = 2 * abs(y.meas[i] - y.pred[i])/(abs(y.meas[i]) + abs(y.pred[i]))
    }
    return(mean(s))
}
best.AIC$sMAPE = rep(0, length(best.AIC[, 1]))
for (i in seq(length(best.AIC[, 1]))) {
    modfit = arima(ts.train, order = as.vector(as.numeric(best.AIC[i, 1:3])),
        seasonal = list(order = as.vector(as.numeric(best.AIC[i, 4:6])),
            period = as.numeric(best.AIC[i, 7])), method = "ML")
    pr = predict(modfit, n.ahead = 12)
    best.AIC[i, 11] = sMAPE(ts.dev, pr$pred)
```

```
}
best.AIC[order(best.AIC$sMAPE), ]
```

```
##    p d q P D Q  m    AIC Box.p Box.p.sq   sMAPE
## 13 1 1 1 0 1 1 12 -117.6 0.524    0.973 0.01749
## 15 1 1 1 1 1 1 12 -118.4 0.635    0.953 0.01831
## 16 2 1 1 1 1 1 12 -121.4 0.926    0.989 0.02093
## 14 2 1 1 0 1 1 12 -120.4 0.859    0.984 0.02253
```

2.4 Model Selection:

Of the four models, ARIMA(1,1,1)x(0,1,1)[12] predicts the development set most accurately.

Let's now consider the task at hand: choosing the model that will best predict 2015. For this analysis, we have intentionally withheld 2015 from all modeling to mimic what might happen one year into the future, for which there is no data. We might be tempted to use (1,1,1)x(0,1,1) since it is the most accurately predicts the withheld 2014 data and is the most parsimonious of the four best AICs. However, this model had the worst AIC of the top 4 models, despite having the fewest parameters. This means that the other models fit the 1990-2013 training set well enough to overcome the high-parameter deficit. Furthermore, we have only one set of withheld data to which we're assessing best fit. This means that the (1,1,1)x(0,1,1) model might have out-performed the other three just by chance. To improve the robustness of our out-of-sample test strategy, we should should perform time series cross-validation as descibed by Hyndman (https://robjhyndman.com/hyndsight/tscv/).

3. Selected Model Performance on Test Set:

Although it does not do as well as the ARIMA(1,1,1)x(0,1,1)[12] on the 2014 development set, we will choose ARIMA(2,1,1)x(1,1,1)[12] as the final model. Of the top four AICs, this model that had the most parameters, yet somehow still the best AIC score, which implies it fits the data better than any of the other three models. It also has the highest Ljung-Box p-value, meaning it is the least likely to be auto-correlated in the residuals. Finally, this model most closely matches our initial post-EDA hypothesis, since both the ACF and PACF showed significance around lag (2). Let's now compare the accuracy of our hypothesis on the 2015 test data.

```
par(mfrow = c(1, 2))
ts.tot.train = rbind(ts.train, ts.dev)
best.arima1 = arima(ts.tot.train, order = c(2, 1, 1), seasonal = list(order = c(0,
    1, 1), period = 12), method = "ML")

pred1 = predict(best.arima1, n.ahead = 11)$pred
smape1 = sMAPE(ts.test, pred1)
print(paste("(2,1,1)x(1,1,1): Symmetric MAPE for test set:", round(smape1,
    4)))
```

```
## [1] "(2,1,1)x(1,1,1): Symmetric MAPE for test set: 0.0374"
```

```
best.arima2 = arima(ts.tot.train, order = c(1, 1, 1), seasonal = list(order = c(0,
    1, 1), period = 12), method = "ML")
pred2 = predict(best.arima2, n.ahead = 11)$pred
smape2 = sMAPE(ts.test, pred2)
print(paste("(1,1,1)x(0,1,1): Symmetric MAPE for test set:", round(smape2,
    4)))
```

```
## [1] "(1,1,1)x(0,1,1): Symmetric MAPE for test set: 0.0334"
```