Final Project — Forecasting Monthly "Time-Series" Related Questions on the Stack Exchange

Andrew Mashhadi

Spring 2023

Introduction

Since 2008, the *Stack Exchange* has been one of the most popular networks of question-and-answer websites. The Stack Exchange network consists of over 100 different websites, covering a variety of specific topics and subjects. As of today, the most actively used sites within the Stack Exchange network are Stack Overflow, which consists of computer programming questions, and Mathematics, which consists of mathematics related questions. Questions asked on these websites are usually technical, and are generally about the details of a specific subject under the given category of the Stack Exchange website. The question-and-answer platform can often be an excellent resource for users looking to learn more about very specific topics that may arise during an academic course, at work, or while self-studying. When submitting a question, each Stack Exchange website offers an option to use tags to help other users find the questions they may be interested in answering, and to help future users find questions they may also have. The number of questions with specific tags over time may serve as a useful dataset to researchers looking to characterize the interest of certain topics over time.

Time series analysis is a topic within mathematics and statistics that encompasses techniques for examining or modeling data points arranged in chronological order, with the aim of extracting meaningful statistics and other characteristics from the data. While traditional time series analysis has a long history spanning several decades, the recent surge in machine learning and data science has sparked a renewed interest in this subject. This course, *Stats 415*, also happens to focus primarily on time series analysis and its applications. Therefore, in this study, we aim to analyze and model the monthly count of Stack Exchange questions tagged with "time-series", using a variety of methods and approaches discussed in class. In addition, we would like to use our final model to provide a forecast of the monthly counts of "time-series" related questions asked over the next few years.

Data

As mentioned in the previous section, our data will consist of monthly counts of Stack Exchange questions tagged with "time-series". In other words, the data will be the number of time series related questions for each available month. Fortunately, the Stack Exchange provides an open source tool known as the *Stack Exchange Data Explorer* for running arbitrary SQL queries against public data from the Stack Exchange network [1]. This tool includes collaborative query editing for all graduate and public Stack Exchange websites. The SQL query used for this paper is provided in the Appendix.

The Stack Exchange Data Explorer was used to collect and save all available "time-series" tagged questions. The data consisted of monthly counts from January 2009 to May 2023, with no missing entries. Therefore, our full data set contained exactly $n = 173$ observations. The first 90% of our dataset (Jan. 2009 - Nov. 2021) was used as a designated training set and the remaining 10% (Dec. 2021 - May 2023) was used as an "unseen" testing set. We display the monthly counts of "time-series" tagged questions from our training set in Figure (1). We immediately notice that there is a clear upward trend in the data from 2009 to 2021. This may be due to the increasing interest in time series analysis, however, it could also be due to other factors such as an increase of users on the Stack Exchange network. Additionally, we notice the obvious spike in counts occurring in 2020. This relatively large spike is most likely a result of the COVID-19 pandemic. Time series analysis and modeling were often used to predict or forecast surges in cases or deaths throughout the pandemic.

We also assess the *autocorrelation* (ACF) and the *partial autocorrelation* (PACF) for the raw counts in our training data. The plots shown in Figure (2) indicate that the trend and/or cycles in our data need to be removed before we begin modeling. In particular, we can see from the large ACF values displayed in the top plot of Figure (2) that our data is currently being dominated by the trend. The ACF is larger than the upper bound of the confidence band for the first 20 lag values. Assuming our data is trend stationary, the strong trend in our data will most likely obscure the behavior of the associated stationary process and prevent us from successfully fitting any *autoregressive moving average* (ARMA) models. The strong trend would also introduce extremely low frequency components in a periodogram that may obscure the appearance at higher frequencies and would most likely prevent us from accurately performing a spectral analysis.

Removing Trend

If we assume our model is trend stationary, our original observations, x_t, are composed of a trend, μ_t, and a stationary process, y_t, such that:

$$x_t = \mu_t + y_t$$

for $t = 1, 2, ..., 173$. As discussed at the end of the previous section, the trend, μ_t, must be removed before we can conduct much of our analysis and modeling. In this section, we discuss the steps involved to obtain a reasonable estimate of the trend component, $\hat{\mu}_t$, and then work with the corresponding residuals

$$\hat{y}_t = x_t - \hat{\mu}_t$$

in our remaining analysis and modeling sections.

From Figure (1), we can see that the relationship between the time and the monthly counts does appear non-linear, but a linear fit may be able capture the majority of the trend. Therefore, we first fit the training data using linear regression, and then we fit the training data with a third-order polynomial. Using an ANOVA table, we rejected the hypothesis that the linear fit was sufficient, and found that the third-order polynomial had a significantly better fit of the data (see Table (1)). That is, we found that the drop in the RSS was effectively "worth" the cost of two additional degrees of freedom. The polynomial fit's F-test resulted in a p-value close to 0, and the R-Squared was approximately 0.92, suggesting that the third-order polynomial fits the training data very well and is able to explain a large proportion of the variance (see Table (2)). Visually, we can see the third-order polynomial was successfully fit to the training data in Figure (3). Therefore, using the computed coefficients from the polynomial fit, our final estimate of the trend is:

$$\hat{\mu}_t = 78.95 + 651.301t - 16.89t^2 - 66.86t^3$$

Now that we have found an acceptable fit for our trend, we subtract our fitted values, $\hat{\mu}_t$, from the raw counts, x_t, to obtain our detrended residuals, $\hat{y}_t$. We can see from Figure (4) that the detrended residuals, $\hat{y}_t$, resembles the stationary time series that we desire.

Spectral Analysis

Now that we have successfully removed the trend from our training data, we investigate the fluctuations in the detrended counts in search of obvious cycles. To do this, we estimate the spectrum of our detrended training data to help us determine predominant periods and to obtain confidence intervals for the identified periods. In Figure (5), we display the periodogram without smoothing (top) and the periodogram with nonparametric smoothing (bottom). The *modified Daniell kernel* with $L = 5$ and a 20% taper was used for the nonparameteric smoothing method. In addition, we separately fit the parametric (autoregressive) spectral estimator shown in the bottom of Figure (6). To fit the autoregressive spectral estimator, we found the AR model that minimized the training *root mean squared error* (RMSE). We can see from the top plot of Figure (6) that the AR(15) model minimized the training RMSE, so we believe it provided the best fit AR model. Therefore, the spectrum of the fitted AR(15) model was used to provide the parametric spectral estimator.

From all three spectral estimators shown in Figures (5) & (6), the frequency axis is in cycles per year (in this case, `frequency` $* 12$, because data points were taken monthly). Additionally, all three spectral estimations illustrate an obvious peak at $\omega \approx 1 \cdot \frac{1}{12} = \frac{1}{12}$ (1 year), along with two other potentially significant peaks at $\omega \approx \frac{3}{10} \cdot \frac{1}{12} = \frac{1}{40}$ (40 months) and $\omega \approx 3 \cdot \frac{1}{12} = \frac{1}{4}$ (4 months). Using the periodogram with nonparametric smoothing, we generated the approximate one-sided and two-sided 95% confidence intervals for the true spectral power at the three frequencies of interest (shown in Table (3)).

In all spectral estimates, the narrow peak at frequency $w \approx \frac{1}{12}$ achieves the highest estimated spectral power and represents the annual cyclical pattern clearly displayed in the detrended residuals (Figure (4)). Since the lower bound of the one-sided confidence interval shown in Table (3) is much larger than all other (near-by) periodogram ordinates, we have evidence suggesting that the frequency $\omega \approx \frac{1}{12}$ (1 year) corresponds to a statistically significant peak. This apparent annual cycle may be from academic institutions teaching "time-series" courses during select terms of the school year, leading to more or less questions posted online depending on the time of year. In Figure (7), we illustrate the cyclical behavior occurring in each year by stacking the plots of the detrended monthly counts from 2014, 2015, 2016, and 2017. The annual cycle seen in each plot may be described with the following stages: (1) an increase in values within the first quarter of the year (2) a drop in values between April, May, and June (3) higher values by August (the Summer) (4) extremely low values in September (5) an increase in values up to a relative peak in November and (6) values then drop in December (Winter break).

The peak at $\omega \approx \frac{1}{40}$ is also seen in all three spectral estimates. We can see from the spectral estimations in Figures (5) & (6) and the associated confidence intervals from Table (3) that this peak has a much wider band and does not appear to have significantly large spectral power. The wider band typically suggests that this cycle may be slightly irregular. In fact, only looking at the two nonparameteric spectral estimations may lead us to believe that this 40-month cycle is simply due to random noise in our data. However, the parametric spectral estimation also displays a somewhat large spike in the neighborhood of $\omega \approx \frac{1}{40}$. Since parametric autoregressive spectral estimators generally have superior resolution in problems when several closely spaced narrow spectral peaks are present [2], we believe there is a 40-month cycle in our data.

All three spectral estimates also indicate another small peak at $\omega \approx \frac{1}{4}$. This peak seems to have a wider band than the peak associated with the annual cycle, but not as wide as the peak associated with the 40-month cycle. Although the absolute spectral power is not as large as the other two peaks, the lower bound of the one-sided confidence interval shown in Table (3) is also larger than all other (near-by) values. This indicates that the frequency $\omega \approx \frac{1}{4}$ corresponds to a statistically significant peak. The wider band associated with this peak also suggests that this cycle is irregular, but tends to be around 4 months on average. Such a cycle may be from the general structure of time-series courses taught each academic term. For instance, we may see an increase of online posts when more advanced topics are reached in the later weeks of a typical time series analysis course.

It is important to note that the frequency associated with the 4-month peak is located at a multiple of $\omega \approx \frac{1}{12}$, so it may simply be a *harmonic* of the annual cycle seen earlier. Since we do not see a harmonic at $\omega \approx \frac{2}{12}$, we do not believe this is the case. However, this consideration was made before removing the cycle from our data.

After identifying the three predominant cycles in the detrended training data, we iteratively removed each of them from our data. For each cycle, we first averaged the remaining monthly residuals over the cycle's associated period, and then subtracted the averages from each corresponding data point. In Figures (8)-(10) we display each cycle's associated average values (top) and the residuals remaining after removing the cycle from the data (bottom). After all three observed cycles were effectively removed from the detrended data, we computed the ACF and PACF from our remaining, *final*, residuals (Figure (11)). We see that the ACF and PACF values are no longer dominated by trend or seasonality, indicating that the data may now be appropriately used with our models.

Modeling

Now that we have removed both trend and cycles from our training data, we proceed to fit our ARMA models. In each ARMA model, the order of the autoregressive (AR) and moving average (MA) processes are represented by p and q, respectively. As depicted in Figure (11), the ACF and PACF values do not exhibit a definitive cutoff at a specific lag. This observation suggests that our final residual data may conform to an ARMA model (such that $p > 0$ and $q > 0$). To explore the potential success of different models comprehensively, we conduct a grid search to tune our ARMA model and identify the optimal parameters within the ranges of $p = 0, 1, ..., 11$ and $q = 0, 1, ..., 11$.

We evaluated a total of 144 ARMA(p, q) models fit to the data, and calculated the *Akaike Information Criterion* (AIC) for each model. The AIC served as a criterion to identify the optimal ARMA model by determining the combination of p and q that minimized the AIC. To visually compare the AIC values associated with each model, we generated Figure (12). By examining the plot, we observed that while a few other models displayed similar AIC scores, the ARMA(3, 8) model achieved the lowest AIC score of 1197.51 (represented by the red dot in Figure (12)). Consequently, we conclude that the ARMA(3, 8) model provides the best fit to the final residuals obtained from the training data. Using the estimated coefficients, the fitted ARMA(3, 8) model may be written in its complete mathematical form as follows:

$$\begin{aligned} r_t = 0.107 - 0.407r_{t-1} + 0.470r_{t-2} + 0.631r_{t-3} + w_t + 0.763w_{t-1} - 0.279w_{t-2} \\ - 0.754w_{t-3} - 0.060w_{t-4} + 0.030w_{t-5} - 0.073w_{t-6} - 0.334w_{t-7} - 0.311w_{t-8} \end{aligned}$$

where r_t represents our time series of interest (the final residuals) and $w_t \sim wn(0, \sigma_w^2)$. Notice from the equation that the value of the final residual at time t, r_t, is positively related with the values at $t-2$ and $t-3$, but it is negatively related to the previous value at $t-1$. That is, with all else being equal, large positive values observed 2 or 3 months in the past will generally lead to an increase in the predicted value for the current month, but a large positive value observed in the previous month will generally decrease the predicted value for the current month.

Subsequently, we proceeded to generate diagnostic plots for the residuals of our selected optimal model. These plots include the standardized residuals over time, the ACF of the residuals, a normal Q-Q plot of the standardized residuals, and the p-values associated with the Q-statistic (as depicted in Figure (13)). Upon analyzing the time plot of the standardized residuals, no discernible patterns were observed, and only a single outlier was seen surpassing 3 standard deviations in magnitude during 2020. The ACF plot of the standardized residuals displayed no significant deviations from the model assumptions, and the Q-statistic remained non-significant across the depicted lags. The normal Q-Q plot of the residuals indicated a reasonable adherence to the assumption of normality, albeit with a couple potential outliers. Additionally, we computed the spectrum of the residuals from our fitted ARMA(3, 8) model, as illustrated in Figure (14). We can see that there are no significantly large, or outlying, spikes within the estimated spectrum, suggesting that there are no obvious cycles remaining in our model's residuals. Therefore, we conclude that our ARMA(3, 8) model was able to successfully fit the detrended and cycle-removed data.

Predictions & Forecasting

Based on the residual analysis, the model appears to adequately fit the training data with trend and cycles removed. However, we also want to examine our model's overall fit of the original data, in raw counts of "time-series" questions. Furthermore, we would like to assess how well our model performs on the completely unseen test data. Therefore, we generated predictions from our ARMA(3, 8) model for both the training data and the testing data, then we added the associated trend and cycles back into these values to obtain our final predictions in units of monthly counts of "time-series" questions. We should note that the predictions associated with the test set were generated using an 18-step *long-term forecast horizon*. That is, only the data from January 2009 to November 2021 was used to generate the predictions from December 2021 to May 2023 (the test data). Figure (15) displays the predictions (red) overlayed on the original monthly counts from the training and testing data (black). The shaded blue region beginning in 2021 represents

the predictions corresponding to the unseen test data. Although the predictions generally seem to line up quite nicely with the true data, the predictions for the test data certainly appear to have a slightly larger error than the predictions for the training data. In fact, the RMSE for the testing data was found to be 23.25 counts, while the RMSE for the training data was found to be 10.34 counts. Considering that the number of the "time-series" questions are within the range of roughly 0-200 counts, these are both relatively good margins of error to achieve. Our model may be struggling to accurately predict the monthly counts due to the disruption that COVID-19 had on the interest in "time-series" related topics beginning in 2020. However, it is not uncommon for testing error to be larger than the training error. In general, the overall training and testing performances still indicate that our chosen model is demonstrating an excellent fit and good predictive power for future counts of "time-series" data.

To wrap up this section, we proceed to retrain our model using all available data and generate forecasts extending up to May 2026. Our decision to retrain the model with the testing data included was driven by the objective of maximizing data utilization and ensuring the model's up-to-dateness prior to forecasting. In this case, we used another long-term, 36-step, forecast horizon when generating our predictions. Figure (16) presents the raw counts (black), the fitted values (red), and the generated forecasts (blue), accompanied by the associated ± 1 standard error ($S.E.$) depicted by the gray shading. The forecasted trend indicates that the number of questions tagged as "time-series" will reach a plateau throughout 2023 and will follow by a gradual decline leading up to May 2026.

Conclusion

In this paper, we conducted an analysis of monthly counts of Stack Exchange questions tagged with "time-series" using various methods and approaches discussed in the course. We started by obtaining the data from the Stack Exchange Data Explorer and preprocessing it by removing the trend. We then performed spectral analysis to identify obvious cycles in the detrended data, finding an annual cycle, a 40-month cycle, and a 4-month cycle. After removing these cycles, we obtained the final residuals and proceeded to fit autoregressive moving average (ARMA) models. Based on the Akaike Information Criterion (AIC), the ARMA(3, 8) model was selected as the best fit ARMA model.

We assessed our fitted model's performance (with and without the cycles and trend included) using a variety of plots, diagnostics, tests, and performance metrics. The RMSE scores reported for the training and testing data were 10.34 counts and 23.25 counts, respectively. These scores, coupled with the generated figures discussed earlier, indicated an excellent fit of our model and reasonable predictive capability for future counts of "time-series" tagged questions. Moreover, we utilized our final model to forecast the monthly occurrence of "time-series" questions until May 2026. According to our 3-year forecast, the number of questions tagged with "time-series" will eventually reach a plateau within the next year and will then gradually decline until May 2026. Overall, our analysis provided insights into the time-dependent behavior of "time-series" tagged questions on the Stack Exchange and offers a forecasting model for future monthly counts.

Due to the monthly nature of our dataset and the relatively short existence of the Stack Exchange platform for approximately 15 years, our available data is limited to just $n = 173$ observations, which may be considered insufficient by current standards. To enhance future studies, it would be beneficial to explore additional data sources with a longer time span or higher frequency of collection (e.g., daily or weekly). Moreover, it is evident that the COVID-19 pandemic had a clear impact on the monthly counts of "time-series" tagged questions in 2020, which seemingly influenced the subsequent years as well. Therefore, when sufficient time has passed, it would be advantageous to conduct a similar analysis specifically for the period *after* 2020 in order to compare it with our existing model. Lastly, it should be noted that the slight downward trend observed in the previous section's forecasts may be attributed to the polynomial fit of the trend. Polynomial fits typically struggle with extrapolation, as they tend to diverge rapidly when extending beyond the observed time range. Hence, future work should also consider alternative methods for trend fitting to mitigate this limitation.

References

[1] Stack Exchange. *Stack Exchange Data Explorer*. URL: `https://data.stackexchange.com/stackoverflow/query/new`. (accessed: 05.25.2023).

[2] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications With R Examples*. 4th ed. Springer, 2010.

Appendix

Figures

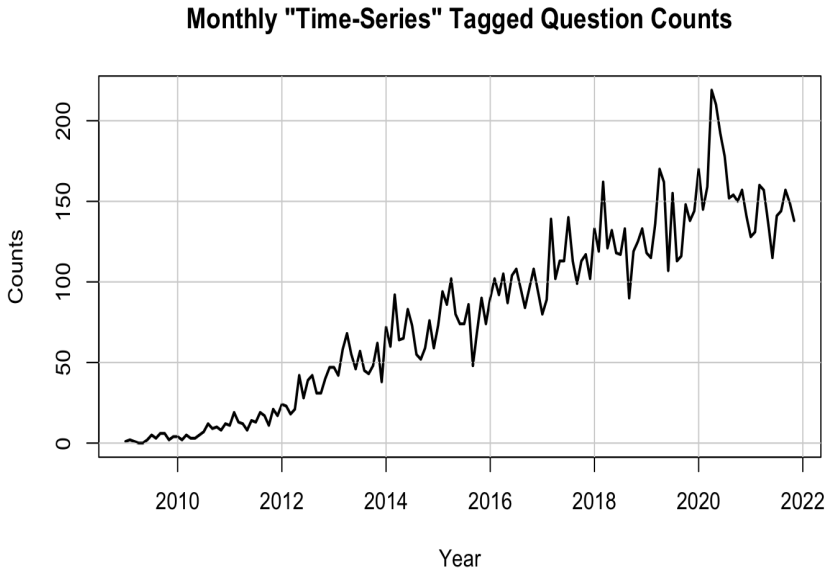

Figure 1: Raw Counts From Training Data

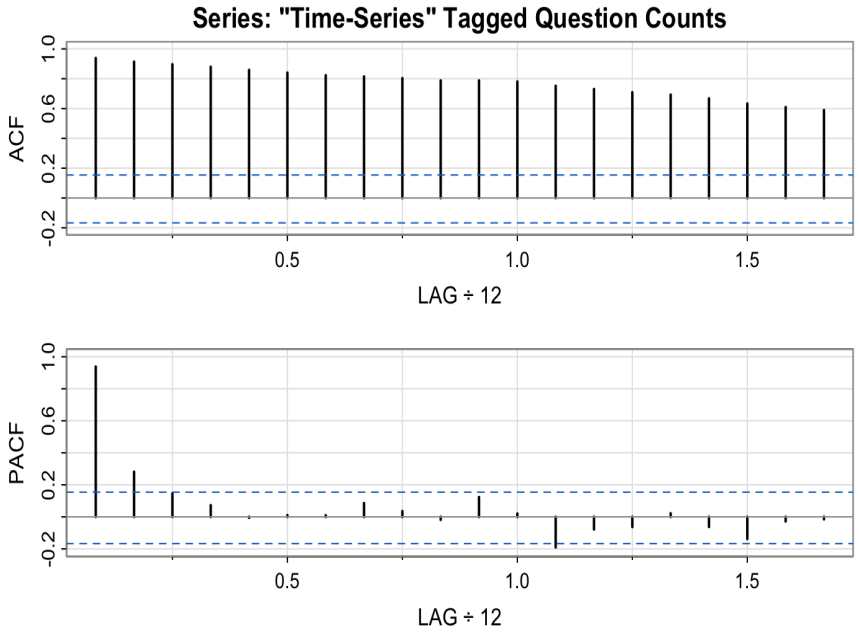

Figure 2: ACF and PACF for Raw Counts From Training Data

Fit	Resid. Df	RSS	DF	Sum. Sq.	F	$\Pr(>F)$
Linear Fit	153	42070				
Poly Deg. 3	151	37315	2	4755.2	9.621	< 0.001

Table 1: Analysis of Variance Table

	Estimate	Std. Error	t value	$\Pr(>\|t\|)$
`(Intercept)`	78.948	1.263	62.525	≈ 0
`time`	651.295	15.720	41.431	≈ 0
`time`^{2}	-16.894	15.720	-1.075	0.284
`time`^{3}	-66.857	15.720	-4.253	≈ 0
Residual standard error: 15.72 on 151 *df* Multiple R-squared: 0.92 F-statistic: 578.6 on 3 and 151 *df* p-value: ≈ 0				

Table 2: Third-order Polynomial Regression Summary

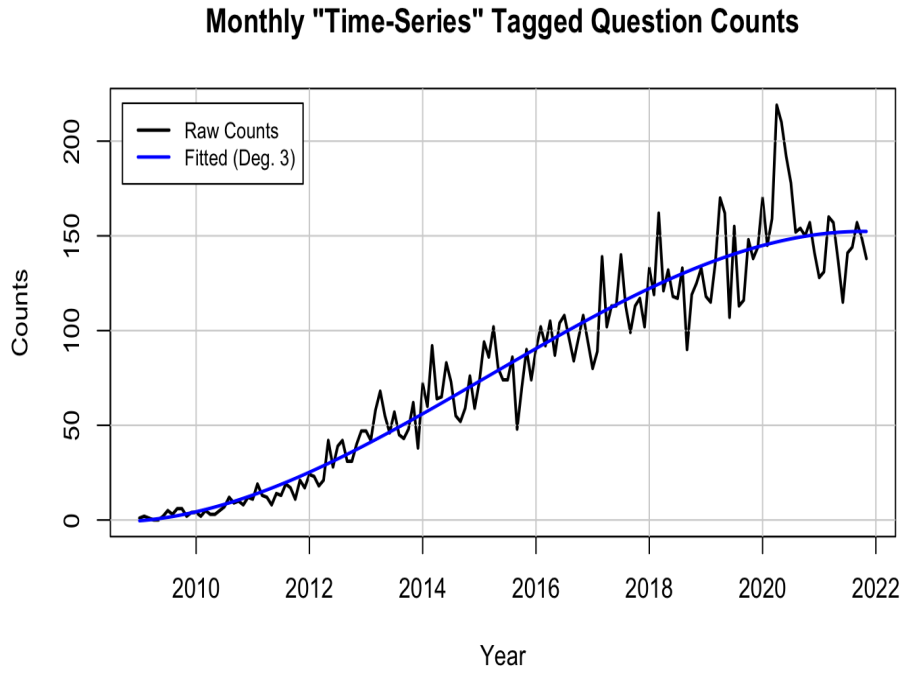

Figure 3: Third-Order Polynomial Fit Over Raw Counts

95% C.I.	Freq. $\times$ 12	Lower	Upper
Two-Sided	≈ 0.3	18.68	96.92
Two-Sided	≈ 1	38.08	197.61
Two-Sided	≈ 3	19.26	99.95
One-Sided	≈ 0.3	20.71	∞
One-Sided	≈ 1	42.23	∞
One-Sided	≈ 3	21.36	∞

Table 3: Non-Parametric Spectral Estimation Confidence Intervals

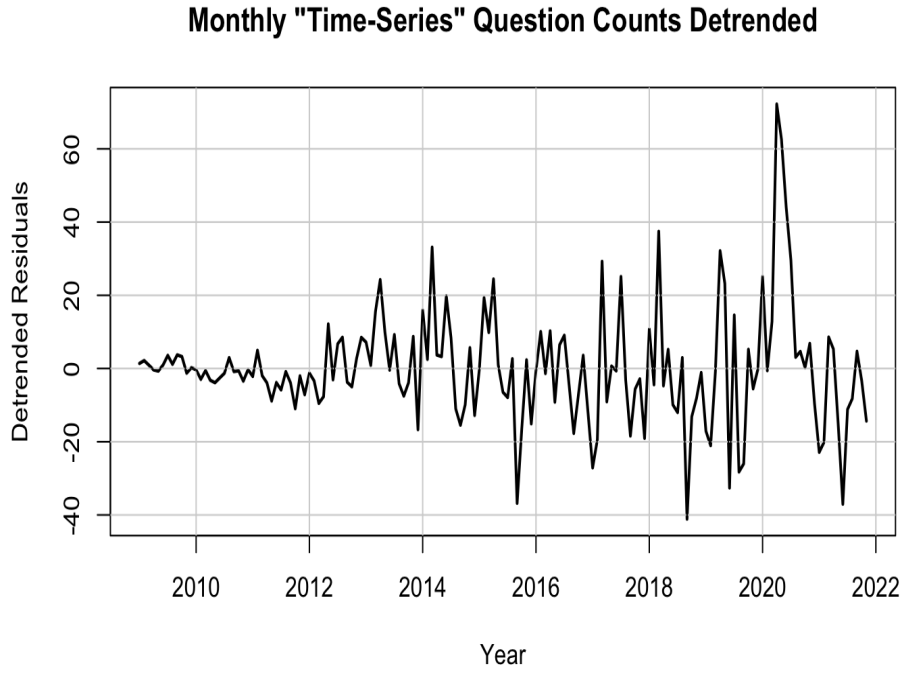

Figure 4: Detrended Residuals (Using Deg. 3 Polynomial for $\hat{\mu}_t$)

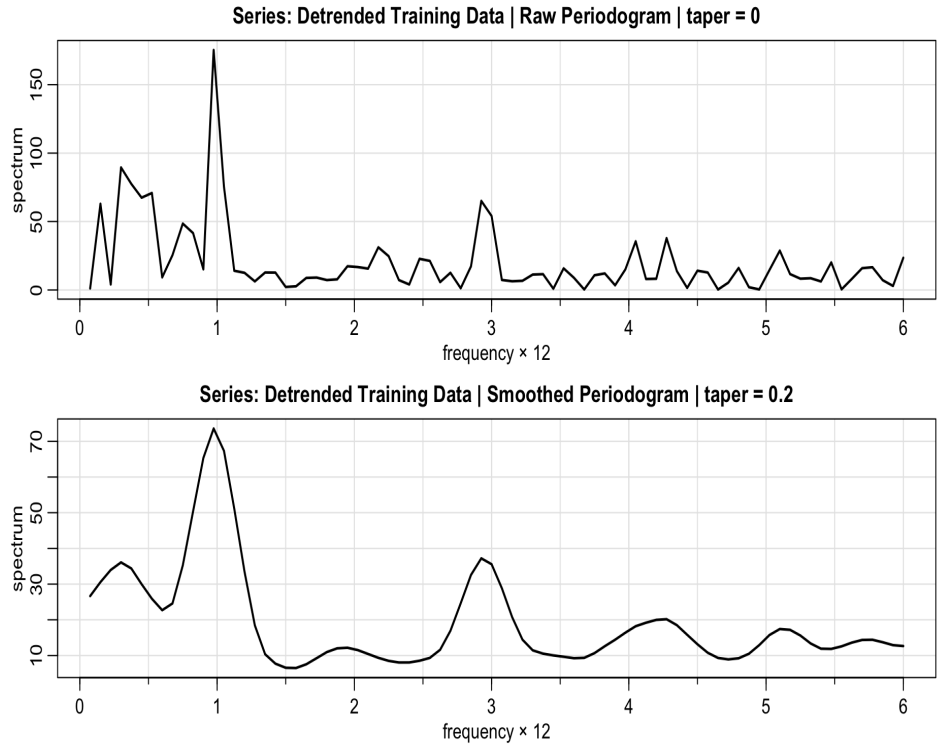

Figure 5: Spectral Estimates for Detrended Residuals

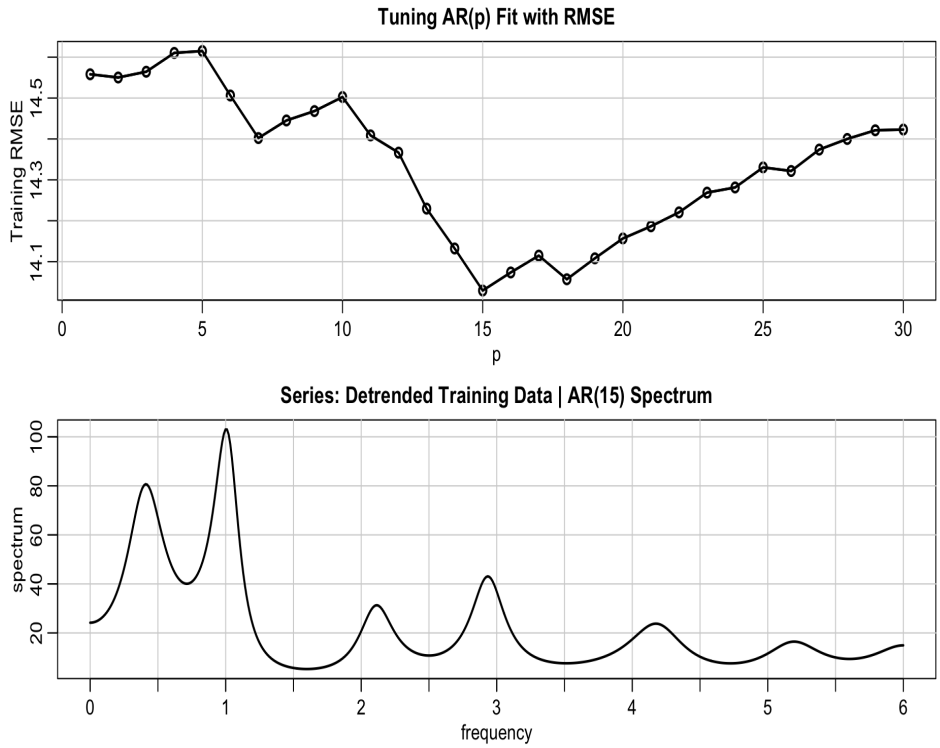

Figure 6: AR Spectral Estimation for Detrended Residuals

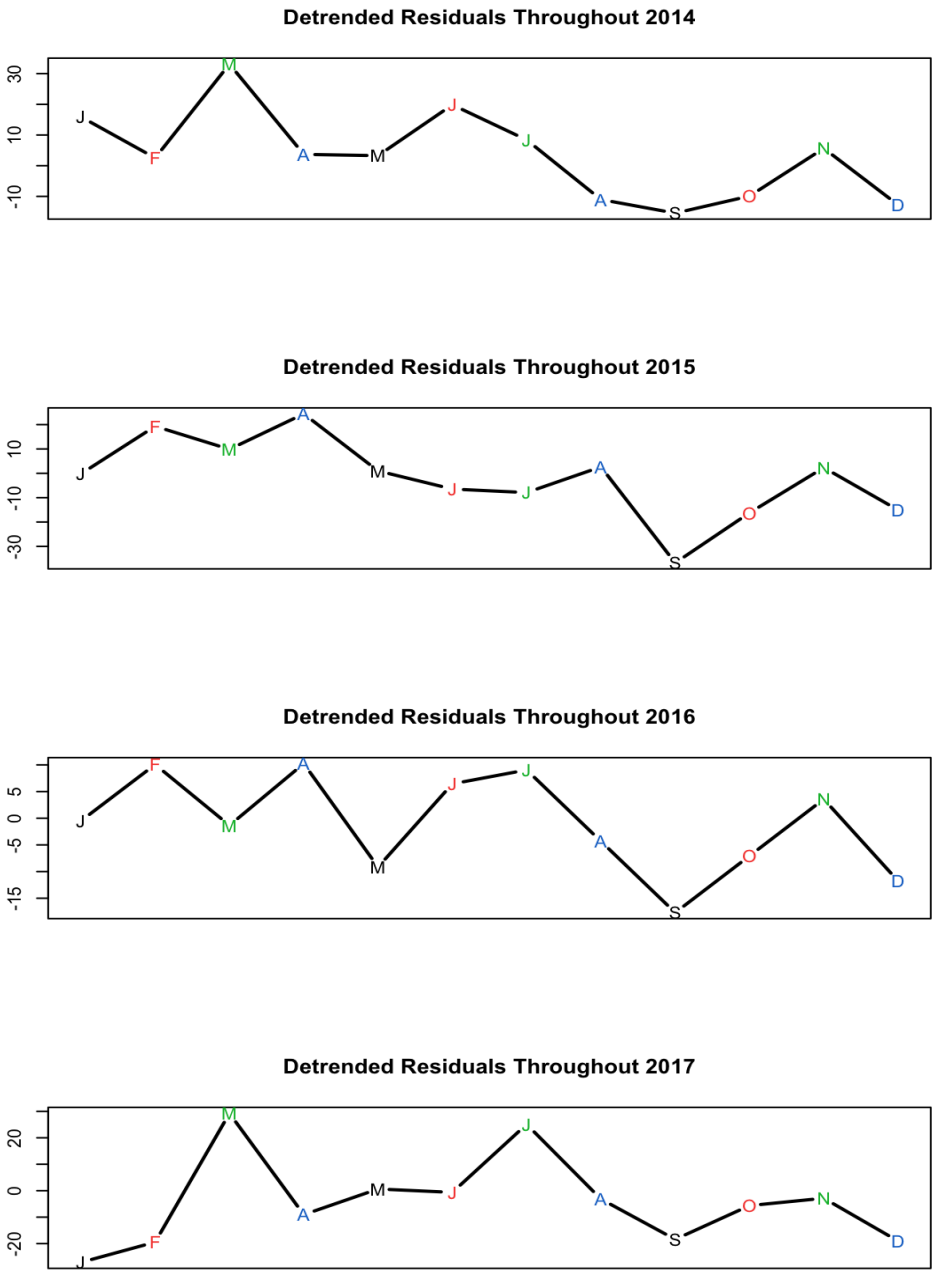

Figure 7: Annual Cycles for 2014 to 2017

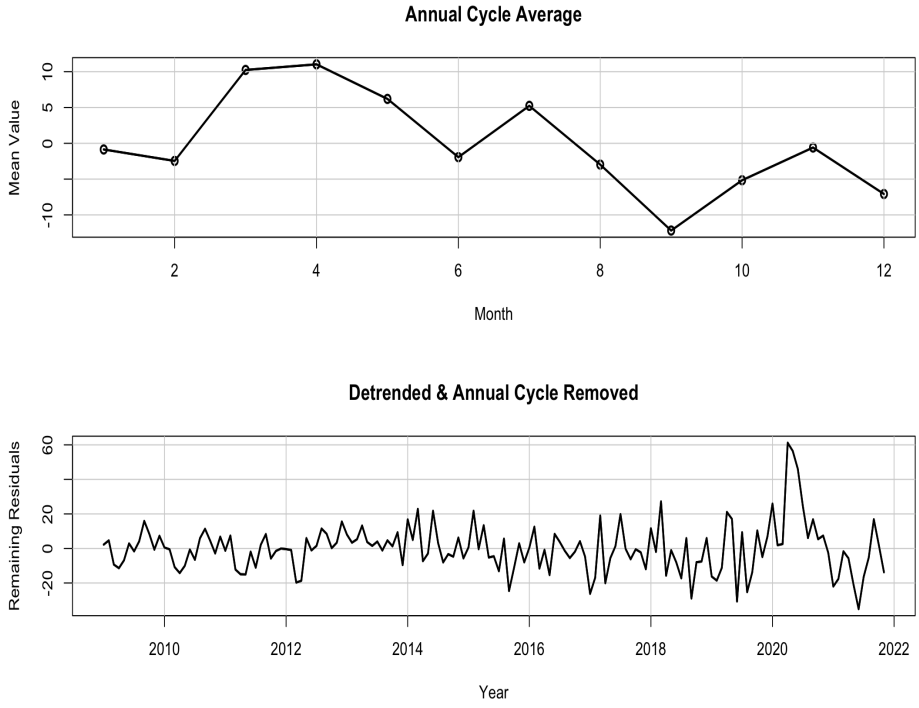

Figure 8: Annual Cycle's Average Values (top) & Remaining Residuals (bottom)

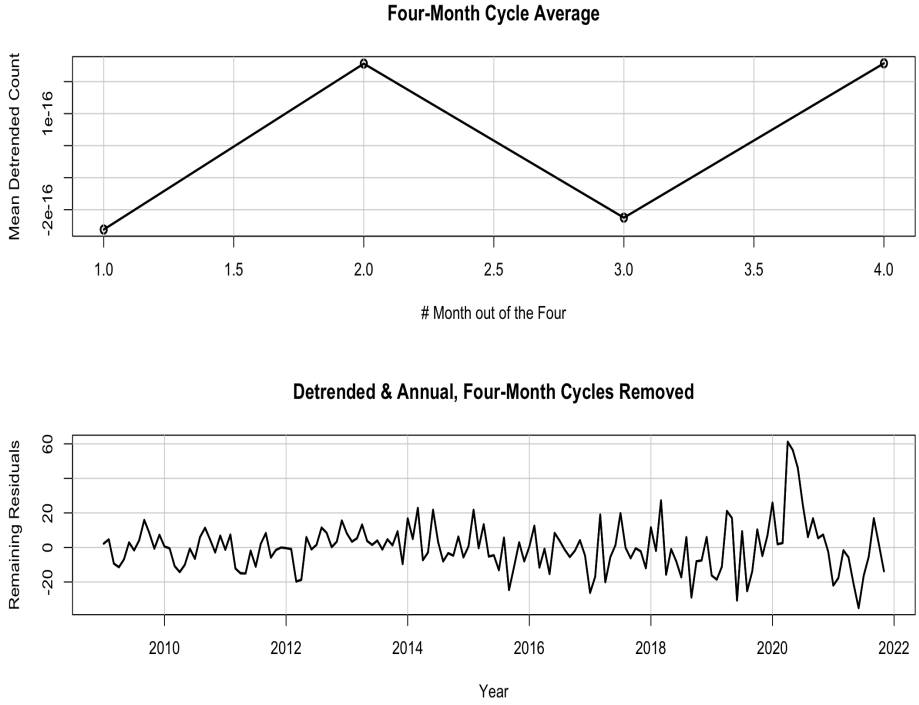

Figure 9: Four-month Cycle's Average Values (top) & Remaining Residuals (bottom)

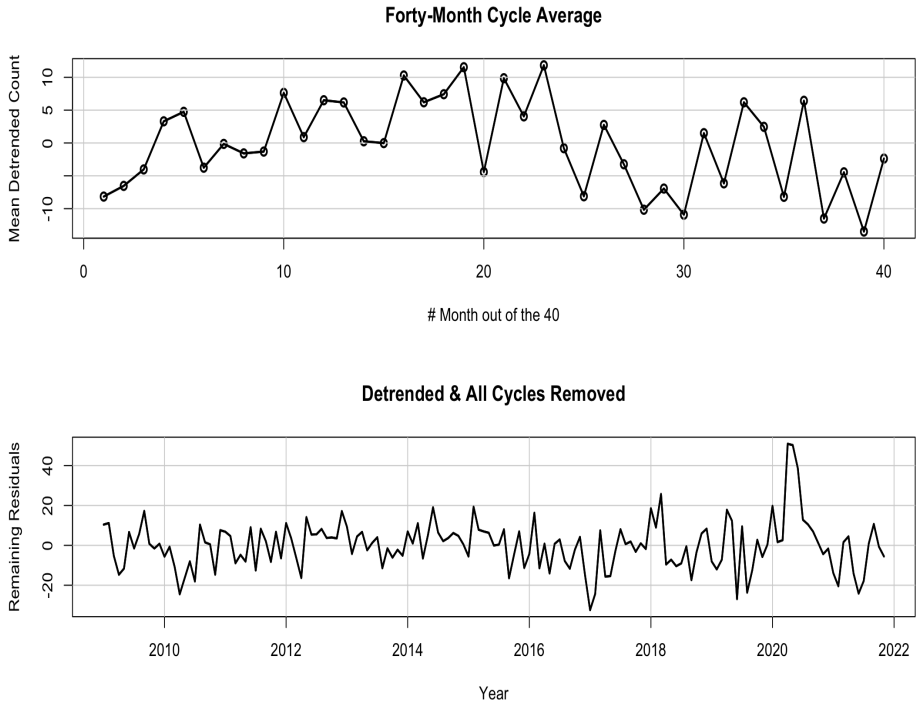

Figure 10: Forty-month Cycle's Average Values (top) & Final Residuals (bottom)

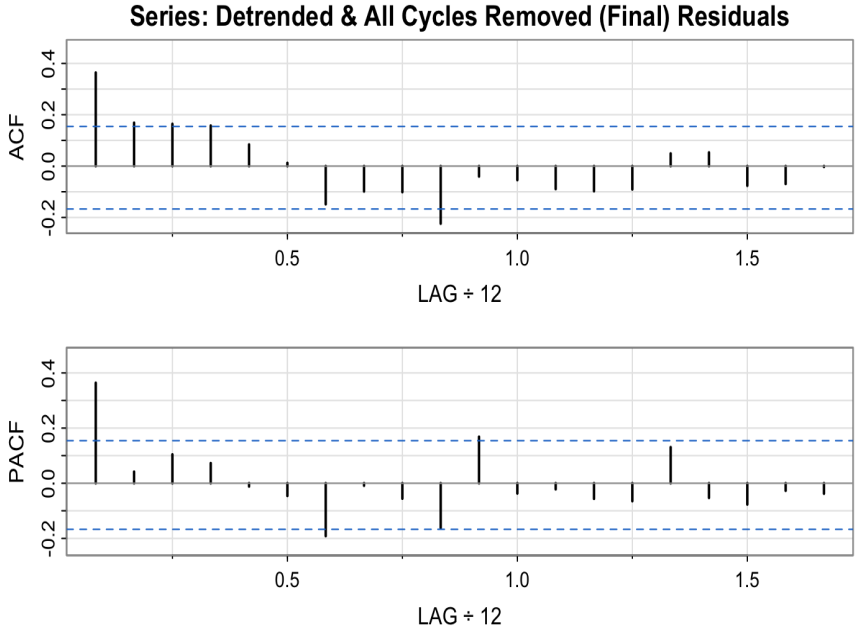

Figure 11: ACF and PACF of the Detrended and Cycle-Removed Residuals

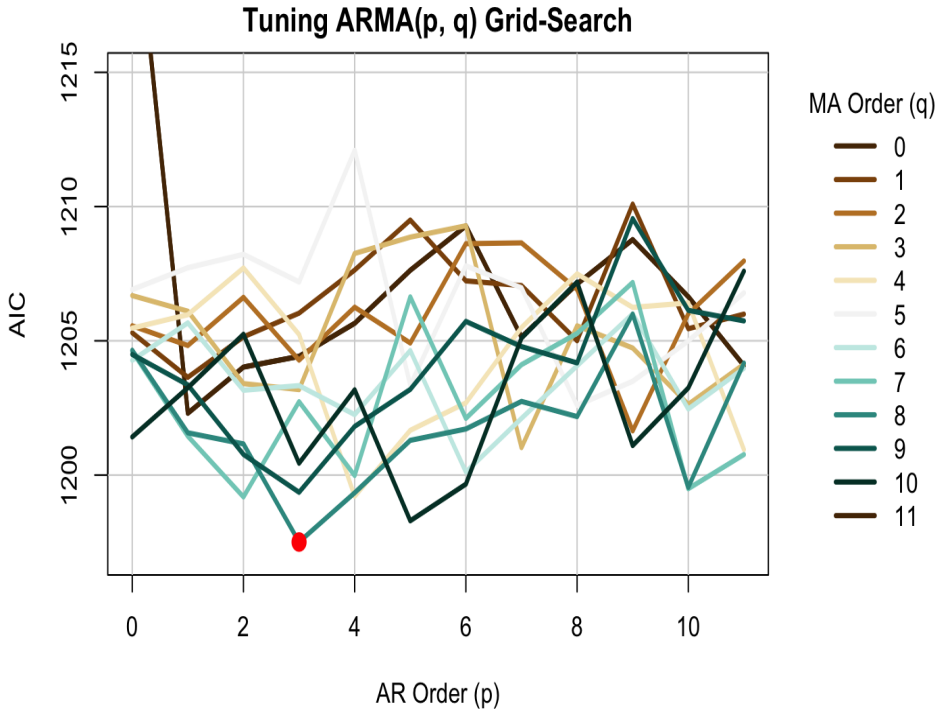

Figure 12: AIC Values for Each ARMA(p, q) Model

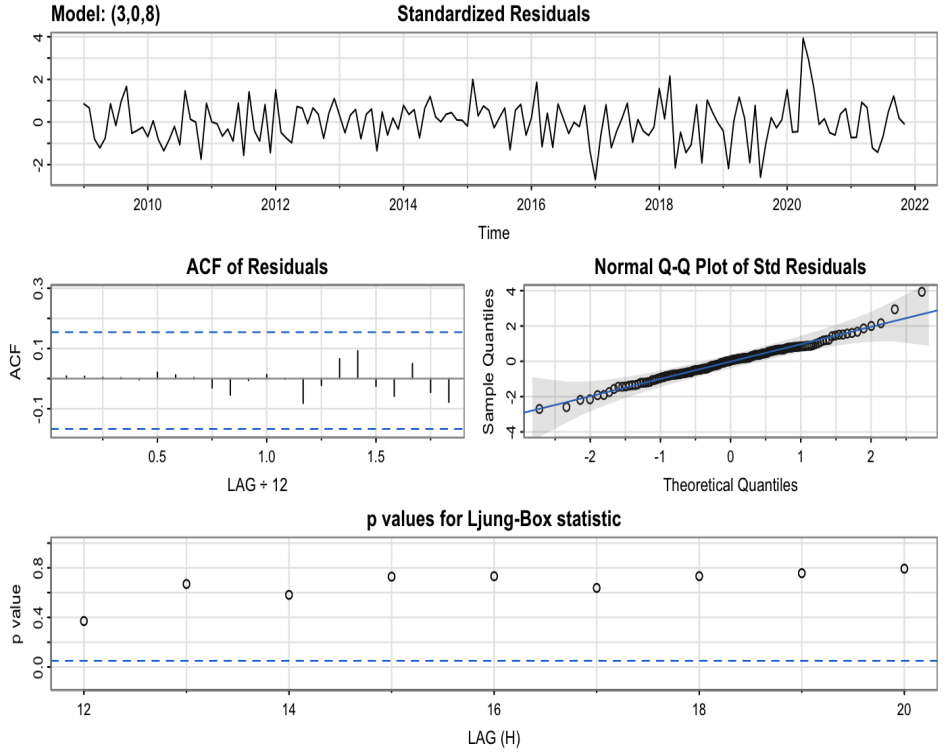

Figure 13: Diagnostic Plots for Fitted ARMA(3, 8) Model

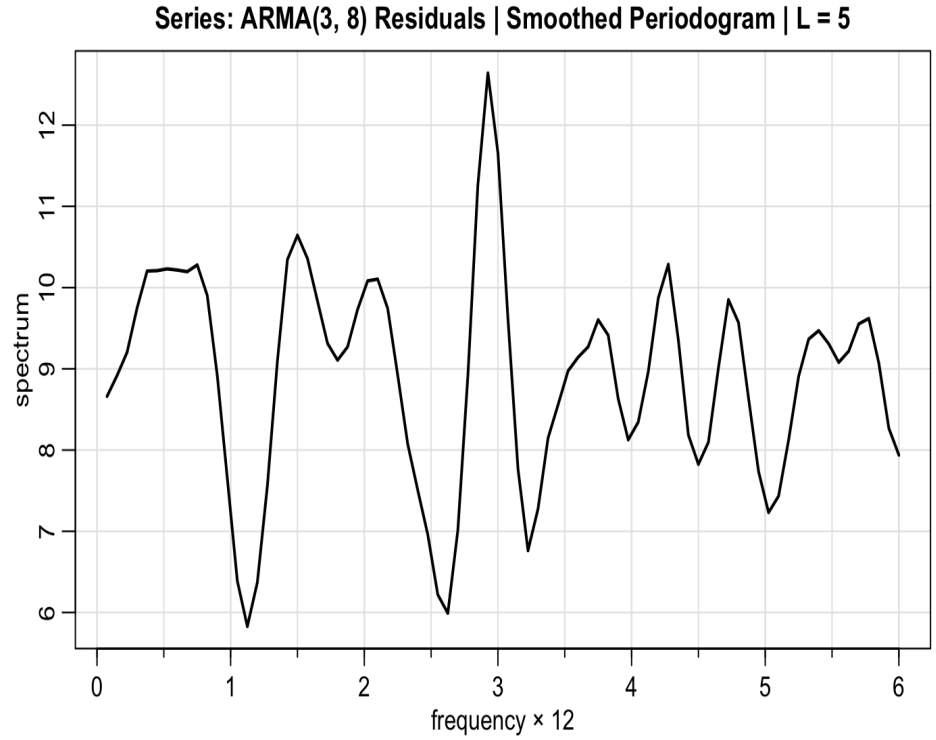

Figure 14: Spectrum for Fitted ARMA(3, 8) Residuals

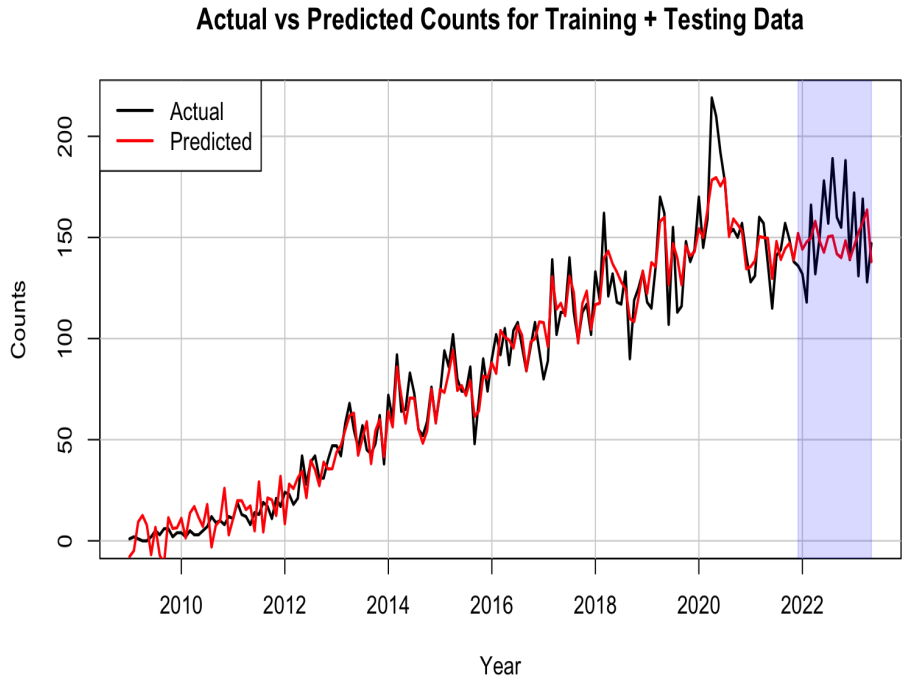

Figure 15: Training and Testing Predictions Over Original Counts

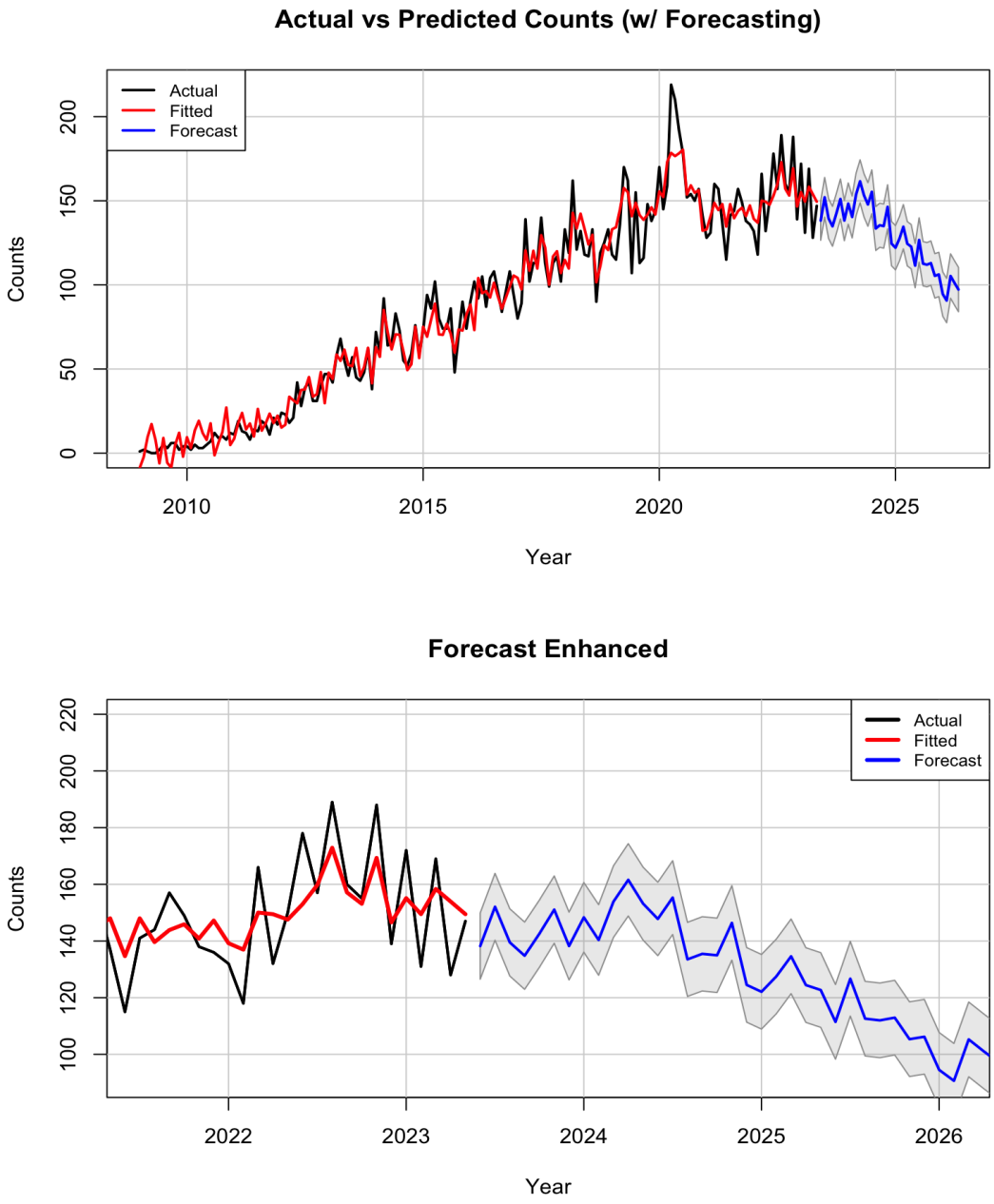

Figure 16: Monthly Counts with Fitted Values and Forecasts

Code

To extract the monthly "time-series" related questions from the Stack Exchange Data Explorer, we used the SQL query shown below:

```sql
SELECT Tags.TagName, year(Posts.CreationDate) AS year, month(Posts.CreationDate) AS month,
    count(Tags.TagName) AS tagcount
FROM Tags
INNER JOIN Posttags ON Tags.Id = Posttags.TagId
INNER JOIN Posts ON Posts.Id = Posttags.PostId
WHERE Tags.TagName = 'time-series'
GROUP BY year(Posts.CreationDate), month(Posts.CreationDate), Tags.TagName
ORDER BY year(Posts.CreationDate), month(Posts.CreationDate)
```

We include the R code used for our analysis and modeling below:

```r

#####################################################################
## load libraries
#####################################################################

library(MASS)
library(astsa)

#####################################################################
## loading data
#####################################################################

# import data
df <- read.csv('dataset/stackexchangecounts_ts.csv')
df <- df[-c(1, 2, nrow(df)), ] # remove incomplete 2008

# first few rows of data
head(df)

# summary of data
summary(df)

# set topic to time-series
count.dat <- df$tagcount

# create time series objects for train/test split
xt <- ts(count.dat, start = c(2009, 1), frequency = 12)

trn <- count.dat[1:floor(0.9*length(count.dat))]
tst <- count.dat[(floor(0.9*length(count.dat))+1):length(count.dat)]
xt_trn <- ts(trn, start = c(2009, 1), frequency = 12)
xt_tst <- ts(tst, start = c(2021, 12), frequency = 12)

# create time plot of training data
ts.plot(xt_trn,
        type = "l",
        lwd = 2,
        main = "Monthly \"Time-Series\" Tagged Question Counts",
        xlab = "Year",
        ylab = "Counts")
grid(lty="solid")
lines(xt_trn)


# acf and pacf, before trend removed
acf2(xt_trn, max.lag = 20, lwd = 2,
     main="Series: \"Time-Series\" Tagged Question Counts")

#####################################################################
## removing trend
```

```
#####################################################################

# fit linear and poly degree 3
xt_trn_df <- data.frame(time=as.vector(time(xt_trn)), count=as.vector(xt_trn))
lm.fit <- lm(count ~ time, data=xt_trn_df) # linear
lm.fit.d3 <- lm(count ~ poly(time, 3), data=xt_trn_df) # poly degree 3

# summarys
summary(lm.fit)
summary(lm.fit.d3)

# compare RSS
anova(lm.fit, lm.fit.d3)

ts.plot(xt_trn,
        type = "l",
        main = "Monthly \"Time-Series\" Tagged Question Counts",
        xlab = "Year",
        ylab = "Counts",
        lwd=2)
grid(lty="solid")
lines(xt_trn)
lines(list(x=time(xt_trn), y=lm.fit.d3$fitted.values),
      lwd=2,
      col="blue")
legend(2008.7, 220,
       legend=c("Raw Counts", "Fitted (Deg. 3)"),
       col=c("black", "blue"),
       lwd=2,
       cex=0.8)

# remove trend relationship, show plot
detrended_xt_trn <- xt_trn - lm.fit.d3$fitted.values
ts.plot(detrended_xt_trn,
        main = "Monthly \"Time-Series\" Question Counts Detrended",
        xlab = "Year",
        ylab = "Detrended Residuals",
        type = "l",
        lwd=2)
grid(lty="solid")
lines(detrended_xt_trn)

#####################################################################
## spectral analysis
#####################################################################

par(mfrow=c(2, 1))

# investigate trend/cycles
dat.spec <- mvspec(detrended_xt_trn, log="no", taper=0,
                   main="Series: Detrended Training Data | Raw Periodogram | taper = 0",
                   lwd=2)

# non-parametric
dat.spec.sm.np <- mvspec(detrended_xt_trn,
                         kernel("modified.daniell", c(2, 2)), # not a large set of data
    points
                         taper=0.2,
                         log="no",
                         main="Series: Detrended Training Data | Smoothed Periodogram |
    taper = 0.2",
                         lwd=2)

# find min ar order that minimizes the training RMSE
n <- length(detrended_xt_trn)
RMSE <- rep(0, 30)
for (k in 1:30){

  resid <- ar(detrended_xt_trn, order=k, aic=FALSE)$resid
```

```
  RMSE[k] <- sqrt(mean(resid^2, na.rm=TRUE))

}

plot(RMSE, type="o", xlab="p", ylab="Training RMSE",
     main="Tuning AR(p) Fit with RMSE",
     lwd=2)
grid(lty="solid")
lines(RMSE, lwd = 2)

# parametric
dat.spec.sm.p <- spec.ar(detrended_xt_trn, order=15, log="no",
                         main="Series: Detrended Training Data | AR(15) Spectrum")
axis(1, at = seq(0, 6, by=0.5), tck = 1, col = "lightgrey", labels=FALSE, lty = "solid")
axis(2, tck = 1, col = "lightgrey", labels = FALSE, lty = "solid", tick=T)
axis(1, at = seq(0, 6, by=0.5), tick = TRUE, labels=FALSE)
axis(2, tick = TRUE, labels=FALSE)
lines(dat.spec.sm.p$freq, dat.spec.sm.p$spec, lwd=2)

# two-sided confidence intervals
cat("\nTWO-SIDED CONFIDENCE INTERVALS\n")
df <- dat.spec.sm.np$df
U <- qchisq(.025, df)
L <- qchisq(.975, df)
cat("For peak at freq =", dat.spec.sm.np$freq[4],
    "has the approximate two-sided CI: [", df*dat.spec.sm.np$spec[4]/L, ",",
    df*dat.spec.sm.np$spec[4]/U, "]\n")
cat("For peak at freq =", dat.spec.sm.np$freq[13],
    "has the approximate two-sided CI: [", df*dat.spec.sm.np$spec[13]/L, ",",
    df*dat.spec.sm.np$spec[13]/U, "]\n")
cat("For peak at freq =", dat.spec.sm.np$freq[39],
    "has the approximate two-sided CI: [", df*dat.spec.sm.np$spec[39]/L, ",",
    df*dat.spec.sm.np$spec[39]/U, "]\n")


# one-sided confidence intervals
cat("\nONE-SIDED CONFIDENCE INTERVALS\n")
L <- qchisq(.95, df)
cat("For peak at freq =", dat.spec.sm.np$freq[4],
    "has the approximate one-sided CI: [", df*dat.spec.sm.np$spec[4]/L,
    ", inf ]\n")
cat("For peak at freq =", dat.spec.sm.np$freq[13],
    "has the approximate one-sided CI: [", df*dat.spec.sm.np$spec[13]/L,
    ", inf ]\n")
cat("For peak at freq =", dat.spec.sm.np$freq[39],
    "has the approximate one-sided CI: [", df*dat.spec.sm.np$spec[39]/L,
    ", inf ]\n")

## visualize the annual cycle with a few example years

# set up 4x1 block of plots
months <- c("J", "F", "M", "A", "M", "J", "J", "A", "S", "O", "N", "D")
par(mfrow=c(4, 1))

# setup plot
years.f2009 <- 5
t1 <- 12*years.f2009 + 1
t2 <- t1 + 11

plot(time(detrended_xt_trn)[t1:t2], detrended_xt_trn[t1:t2],
     type = "c", xlab = "", ylab = "", xaxt = 'n', lwd=2,
     main = "Detrended Residuals Throughout 2014")
points(time(detrended_xt_trn)[t1:t2], detrended_xt_trn[t1:t2], pch = months, col = 1:4)

# setup plot
years.f2009 <- 6
t1 <- 12*years.f2009 + 1
t2 <- t1 + 11

```

```
plot(time(detrended_xt_trn)[t1:t2], detrended_xt_trn[t1:t2],
     type = "c", xlab = "", ylab = "", xaxt = 'n', lwd=2,
     main = "Detrended Residuals Throughout 2015")
points(time(detrended_xt_trn)[t1:t2], detrended_xt_trn[t1:t2], pch = months, col = 1:4)


# setup plot
years.f2009 <- 7
t1 <- 12*years.f2009 + 1
t2 <- t1 + 11

plot(time(detrended_xt_trn)[t1:t2], detrended_xt_trn[t1:t2],
     type = "c", xlab = "", ylab = "", xaxt = 'n', lwd=2,
     main = "Detrended Residuals Throughout 2016")
points(time(detrended_xt_trn)[t1:t2], detrended_xt_trn[t1:t2], pch = months, col = 1:4)

# setup plot
years.f2009 <- 8
t1 <- 12*years.f2009 + 1
t2 <- t1 + 11

plot(time(detrended_xt_trn)[t1:t2], detrended_xt_trn[t1:t2],
     type = "c", xlab = "", ylab = "", xaxt = 'n', lwd = 2,
     main = "Detrended Residuals Throughout 2017")
points(time(detrended_xt_trn)[t1:t2], detrended_xt_trn[t1:t2], pch = months, col = 1:4)

## visualize the 40 month cycle with a few example years

# set up 3x1 block of plots
par(mfrow=c(3, 1))

ts.plot(detrended_xt_trn[1:40],
        type = "l", xlab = "Month", ylab="Detrended Residuals",
        lwd = 2,
        main = "Detrended Residuals Throughout from 2009-2012")

ts.plot(detrended_xt_trn[41:80],
        type = "l", xlab = "Month", ylab="Detrended Residuals",
        lwd = 2,
        main = "Detrended Residuals Throughout from 2012-2015")

ts.plot(detrended_xt_trn[81:120],
        type = "l", xlab = "Month", ylab="Detrended Residuals",
        lwd = 2,
        main = "Detrended Residuals Throughout from 2015-2019")

#####################################################################
## removing cycles
#####################################################################

par(mfrow=c(2, 1))
ann.matrix <- matrix(detrended_xt_trn, byrow = TRUE, ncol = 12)
ann.matrix[13, 12] <- NA # incomplete year in training cutoff

monthly_avg <- colMeans(ann.matrix, na.rm = TRUE)
plot(1:12, monthly_avg, xlab = "Month", ylab = "Mean Value",
     type = "o",
     lwd=2,
     main="Annual Cycle Average")
grid(lty="solid")
lines(1:12, monthly_avg, lwd = 2)

## Now remove this cycle and show the residuals.
final_xt_trn <- detrended_xt_trn - monthly_avg


ts.plot(final_xt_trn, xlab = "Year",
        ylab = "Remaining Residuals",
        main = "Detrended & Annual Cycle Removed",
```

```
        type = "l",
        lwd=2)
grid(lty="solid")
lines(final_xt_trn, lwd = 2)


par(mfrow=c(1, 1))
# assess new spectrum
mvspec(final_xt_trn,
       kernel("modified.daniell", c(2, 2)), # not a large set of data points
       taper=0.2,
       log="no",
       lwd = 2)


par(mfrow=c(2, 1))
four.month.matrix <- matrix(final_xt_trn, byrow = TRUE, ncol = 4)
four.month.matrix[39, 4] <- NA # incomplete year in training cutoff

four.month.avg <- colMeans(four.month.matrix, na.rm = TRUE)
plot(1:4, four.month.avg,
     xlab = "# Month out of the Four",
     ylab = "Mean Detrended Count",
     main = "Four-Month Cycle Average",
     type = "o",
     lwd=2)
grid(lty="solid")
lines(1:4, four.month.avg, lwd = 2)

## Now remove this cycle and show the residuals.
final_xt_trn <- final_xt_trn - four.month.avg

ts.plot(final_xt_trn, xlab = "Year",
        ylab = "Remaining Residuals",
        main = "Detrended & Annual, Four-Month Cycles Removed",
        type = "l",
        lwd=2)
grid(lty="solid")
lines(final_xt_trn, lwd = 2)


par(mfrow=c(1, 1))
# assess new spectrum
mvspec(final_xt_trn,
       kernel("modified.daniell", c(2, 2)), # not a large set of data points
       taper=0.2,
       log="no",
       lwd = 2)

par(mfrow=c(2, 1))
forty.month.matrix <- matrix(final_xt_trn, byrow = TRUE, ncol = 40)
forty.month.matrix[4, c(36:40)] <- NA # incomplete year in training cutoff

forty.month.avg <- colMeans(forty.month.matrix, na.rm = TRUE)
plot(1:40, forty.month.avg,
     xlab = "# Month out of the 40",
     ylab = "Mean Detrended Count",
     main = "Forty-Month Cycle Average",
     lwd=2,
     type = "o")
grid(lty="solid")
lines(1:40, forty.month.avg, lwd = 2)

## Now remove this cycle and show the residuals.
final_xt_trn <- final_xt_trn - forty.month.avg


ts.plot(final_xt_trn,
        xlab = "Year",
```

```
        ylab = "Remaining Residuals",
        main = "Detrended & All Cycles Removed",
        lwd=2,
        type = "l")
grid(lty="solid")
lines(final_xt_trn, lwd = 2)

par(mfrow=c(1, 1))
# assess new spectrum
mvspec(final_xt_trn,
       kernel("modified.daniell", c(2, 2)), # not a large set of data points
       taper=0.2,
       log="no",
       lwd=2)

## estimate new ACF and PACF

# acf and pacf, before trend removed
acf2(final_xt_trn, max.lag = 20,
     main="Series: Detrended & All Cycles Removed (Final) Residuals",
     lwd = 2)

#####################################################################
## fitting the ARMA model to remaining residuals
#####################################################################

# loop through all 12x12 settings of ARMA(p, q) and calculate the AICs
aics <- matrix(0, nrow = 12, ncol = 12)

for(i in 0:11) {
  for(j in 0:11) {

    if (i == 5 & j == 3){

      # issue with non-stationary for this combination of p and q (for CSS)
      t.arma <- arima(final_xt_trn, order = c(i, 0, j), method = "ML")
      aics[i+1,j+1] <- t.arma$aic

    } else {

      t.arma <- arima(final_xt_trn, order = c(i, 0, j))
      aics[i+1,j+1] <- t.arma$aic
    }

  }
}


# load library for plot colors
library(RColorBrewer)

# setup colors
cols <- brewer.pal(12, "BrBG")


## AIC TUNING PLOT
par(mar=c(5,4,2,7))
plot(0:11, aics[, 1],
     xlab = "AR Order (p)",
     ylab = "AIC",
     main = "Tuning ARMA(p, q) Grid-Search",
     type = "l",
     lwd = 3,
     col = cols[1],
     ylim = c(1197, 1215))
grid(lty="solid")
for (r in 1:12) {

  lines(0:11, aics[, r],
```

```
        lwd = 3,
        col = cols[r],
        type = "l")

}
legend("right", inset = c(-0.26,0), legend = 0:11, xpd = NA,
       title = "MA Order (q)", col = cols, lty = 1, bty = "n", lwd=3)
points(3, min(aics), col="red", lwd=4, pch=19)

# use sarima to get the diagnostics
sarima(final_xt_trn, p = 3, q = 8, d = 0)

# fit arima using regular arima model (same coefficients were found)
arma.mod <- arima(final_xt_trn, order=c(3, 0, 8))

# display coefficients and summary
arma.mod$coef

# load library
library(forecast)

# overlay fitted to final residuals training data
ts.plot(final_xt_trn,
        type = "l",
        main = "Detrended + Cycles Removed Residuals with ARMA(3, 8) Fit",
        xlab = "Year",
        ylab = "Detrended + Cycles Removed Residuals")
grid(lty="solid")
lines(final_xt_trn, lwd=2)
lines(list(x=time(final_xt_trn), y=fitted(arma.mod)),
      lwd=2,
      col="red")
legend("topleft", cex = 1, c("Actual", "Predicted"),
       lty = 1, col = c("black", "red"), lwd=2)


# output AIC and training RMSE for residuals
cat("FOR RESIDUALS\n")
cat("AIC:", arma.mod$aic, " | RMSE:", sqrt(mean(arma.mod$residuals^2)), "\n\n")

# convert back to counts
trn.count.preds <- fitted(arma.mod) + monthly_avg + four.month.avg +
  forty.month.avg + lm.fit.d3$fitted.values

# overlay fitted to counts training data
ts.plot(xt_trn,
        type = "l",
        main = "Actual vs Predicted Counts for Training Data",
        xlab = "Year",
        ylab = "Counts",
        lwd=2)
grid(lty="solid")
lines(xt_trn, lwd=2)
lines(list(x=time(xt_trn), y=trn.count.preds),
      lwd=2,
      col="red")
legend("topleft", cex = 1, c("Actual", "Predicted"),
       lty = 1, col = c("black", "red"), lwd=2)


# output training RMSE for counts
cat("FOR COUNTS\n")
cat("RMSE:", sqrt(mean((xt_trn-trn.count.preds)^2)), "\n\n")

## spectrum of residuals from ARMA(3, 8) model

# assess spectrum residuals from ARMA
arma.res.spec <- mvspec(resid(arma.mod),
                        log="no",
```

```
                        main="Series: ARMA(3, 8) Residuals | Raw Periodogram",
                        lwd=2)

# non-parametric
arma.res.spec.np <- mvspec(resid(arma.mod),
                           kernel("modified.daniell", c(2, 2)),
                           log="no",
                           main="Series: ARMA(3, 8) Residuals | Smoothed Periodogram | L = 5
    ",
                           lwd=2)

#################################################################
## testing performance with trained ARMA(3, 8)
#################################################################

# TRAINING + TESTING COUNTS PLOTS

# convert back to counts (+ cycles + trend)
tot.cnt.preds <- c(fitted(arma.mod), predict(arma.mod, 17)$pred) +
  monthly_avg + four.month.avg + forty.month.avg +
  predict(lm.fit.d3, data.frame(time=as.vector(time(xt))))

# overlay fitted to counts
ts.plot(xt,
        type = "l",
        main = "Actual vs Predicted Counts for Training + Testing Data",
        xlab = "Year",
        ylab = "Counts")
grid(lty="solid")
lines(xt, lwd=2)
lines(list(x=time(xt), y=tot.cnt.preds),
      lwd=2,
      col="red")
rect(xleft=head(time(xt_tst),1), xright=tail(time(xt_tst),1),
     ybottom=par("usr")[3], ytop=par("usr")[4],
     density=NA, col = adjustcolor("blue", alpha = 0.15))
legend("topleft", cex = 1, c("Actual", "Predicted"),
       lty = 1, col = c("black", "red"), lwd=2)


## ONLY TRAINING DATA PERFORMANCE

trnpreds <- tot.cnt.preds[1:(length(tot.cnt.preds)-18)]

# output training RMSE for counts
cat("Training RMSE:", sqrt(mean((xt_trn - trnpreds)^2)), "\n\n")

## ONLY TESTING DATA PERFORMANCE

tstpreds <- tot.cnt.preds[(length(tot.cnt.preds)-17):length(tot.cnt.preds)]

# output testing RMSE for counts
cat("Testing RMSE:", sqrt(mean((xt_tst - tstpreds)^2)), "\n\n")

#################################################################
## making forecasts beyond available data
#################################################################

# refit trend, and then remove
xt_df <- data.frame(time=as.vector(time(xt)), count=as.vector(xt))
t.fit <- lm(count ~ poly(time, 3), data=xt_df) # fit
detrended_xt <- xt - t.fit$fitted.values # remove trend

## remove cycles again

ann.matrix <- matrix(detrended_xt, byrow = TRUE, ncol = 12)
ann.matrix[15, c(6:12)] <- NA # incomplete year 2023
monthly_avg <- colMeans(ann.matrix, na.rm = TRUE)

```

```r
# remove annual cycle
final_xt <- detrended_xt - monthly_avg

four.month.matrix <- matrix(final_xt, byrow = TRUE, ncol = 4)
four.month.matrix[44, c(2:4)] <- NA # incomplete year 2023
four.month.avg <- colMeans(four.month.matrix, na.rm = TRUE)

# remove 4 month cycle
final_xt <- final_xt - four.month.avg

forty.month.matrix <- matrix(final_xt, byrow = TRUE, ncol = 40)
forty.month.matrix[5, c(14:40)] <- NA # incomplete year 2023
forty.month.avg <- colMeans(forty.month.matrix, na.rm = TRUE)

# remove 40 month cycle
final_xt <- final_xt - forty.month.avg


# fit arima using regular arima model (same coefficients were found)
full.arma.mod <- arima(final_xt, order=c(3, 0, 8))

## FORECASTING PLOT

# make forecast
fore <- predict(full.arma.mod, 36)

fore.preds <- ts(start = c(2023, 1), end = c(2026, 5), frequency = 12)
fore.preds[-c(1:5)] <- fore$pred

# generate fitted values
tot.fitted <- fitted(full.arma.mod) +
  monthly_avg + four.month.avg + forty.month.avg +
  fitted(t.fit)

# convert to counts (have to align starting at Jan to recycle with the
# averages correctly)
fore.preds <- fore.preds +
  monthly_avg + four.month.avg + forty.month.avg +
  predict(t.fit, data.frame(time=as.vector(time(fore.preds))))

# remove NAs
fore.preds <- window(fore.preds, start=c(2023, 6))

# for CI
U <- fore.preds + fore$se
L <- fore.preds - fore$se
xx <- c(time(U), rev(time(U)))
yy <- c(L, rev(U))

par(mfrow=c(2, 1))
ts.plot(xt,
        type = "l",
        main = "Actual vs Predicted Counts (w/ Forecasting)",
        xlab = "Year",
        ylab = "Counts",
        xlim=c(2009, 2026.3))
grid(lty="solid")
lines(xt, lwd=2)
lines(list(x=time(xt), y=tot.fitted),
      lwd=2,
      col="red")
polygon(xx, yy, border = 8, col = gray(.6, alpha = .2))
lines(list(x=time(fore.preds), y=fore.preds), lwd=2, col="blue")
legend("topleft", cex = 0.8, c("Actual", "Fitted", "Forecast"),
       lty = 1, col = c("black", "red", "blue"), lwd=2)

ts.plot(xt,
        type = "l",
        xlab = "Year",
```

```
        ylab = "Counts",
        main = "Forecast Enhanced",
        lwd = 2,
        xlim=c(2021.5, 2026.1),
        ylim=c(90, 220))
grid(lty="solid")
lines(xt, lwd = 2)
lines(list(x=time(xt), y=tot.fitted),
      lwd=3,
      col="red")
polygon(xx, yy, border = 8, col = gray(.6, alpha = .2))
lines(list(x=time(fore.preds), y=fore.preds), lwd=2, col="blue")
legend("topright", cex = 0.8, c("Actual", "Fitted", "Forecast"),
       lty = 1, col = c("black", "red", "blue"), lwd=3)
```