6372: Unit 4 Homework

Matt Farrow

HW Instructions

The weekly HW assignments are designed to accomplish two goals for the MSDS student. The first is to provide a series of conceptual and analytical questions so the student can get a feel for their current understanding of the unit. The second goal is to introduce the students to standard functions and routines in R that effectively do the same things that the "Procs" do in SAS.

R and SAS are both wonderful tools and as we go through the assignments, students will begin to recognize very quickly that they both have pros and cons.

The formatting of the HW is as follows:

1. A series of high level questions will be asked with either short answers or simple multiple choice responses.
2. Analytical questions will be provided but a short vignette example of how R functions work for a given topic or method will be given. The student will then be asked a follow up question or two based on the output provided.
3. Thirdly, a new data set will be given to allow the student to gain some experience with a new data set from start to finish.

Solutions to the HW will be provided a day or two after the HW is submitted. It is up to the student to "shore up" any confusion or misunderstanding of a topic. Grading will be based on a combination of correctness, completion, and overall conciseness.

The student may provide their answers in a separate word document. Just make sure that it is easy to follow and that all questions have been addressed for the grader. You are welcome to use R markdown, but it is not required.

Time Series Conceptual questions

1. State the necessary requirements for a time series to be stationary.

For a time series to be stationary, the mean, variance and autocorrelation should not change over time.

2. TRUE or FALSE? If a time series model includes an explanatory variable such as time itself or some other predictor, then the original time series of the response is not stationary.

False

3. What is the major draw back of having serially correlated observations?

Sample averages tend to be further away from the long-run mean.

4. What is the major advantage of having serially correlated observations?

Forcasating with serially correlated observations is more successful.

5. What is the purpose of the Durbin Watson test?

The purpose of the Durbin Watson test is to verify that the residuals from a least-squares regression are not autocorrelated with the residuals of an AR(1) model.

Exercise #1

Simulating time series data to help with ACF and PACF diagnostics.

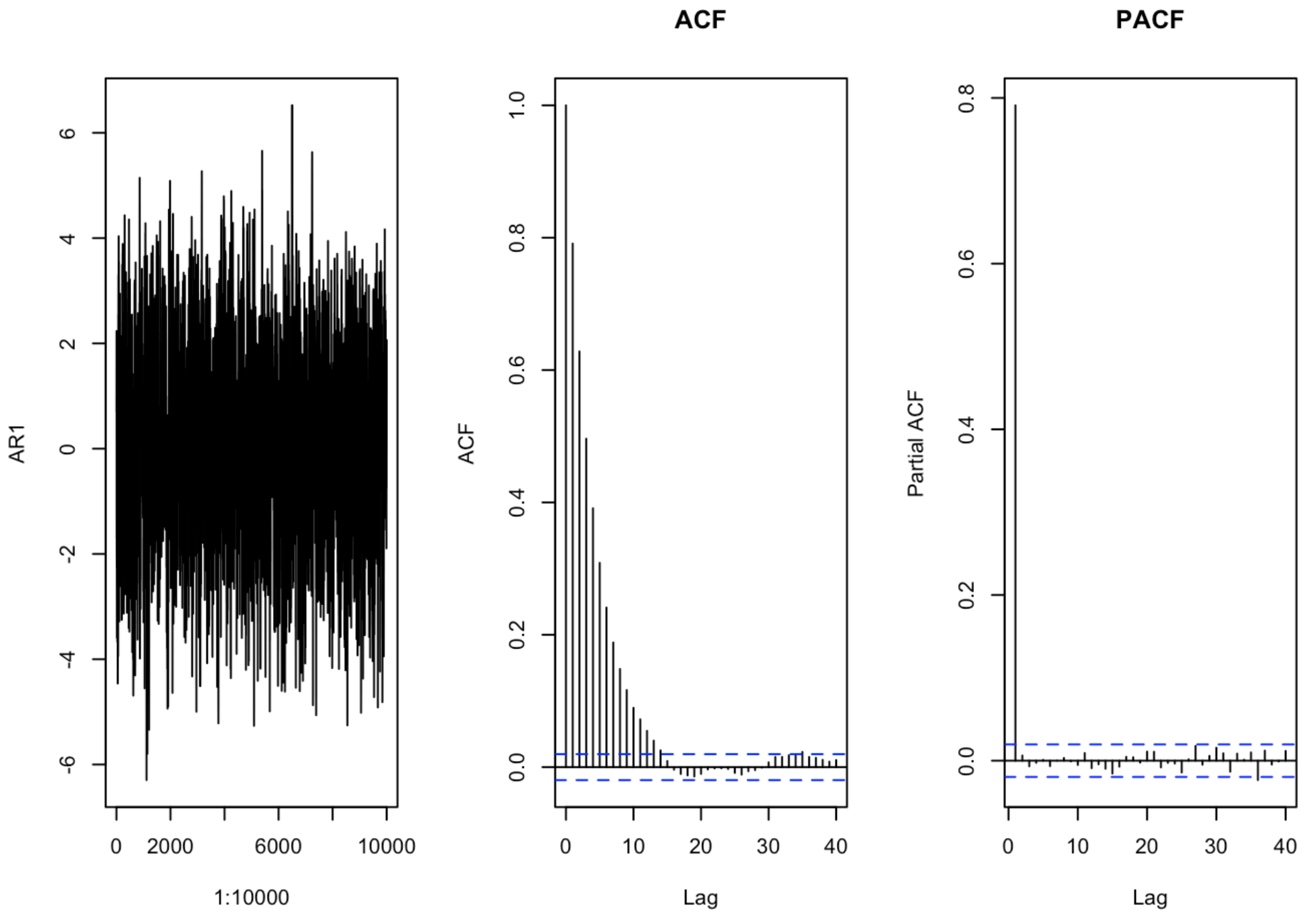
The most common issue that students have with time series is reading/interpreting the ACF and PACF diagnostic plots and using the rules of thumb to help identify what correlation structures should be used. The following scripts are going to provide you with simulated data sets in which we know how the correlation structure truly is (AR(1), AR(2), … etc.). We will then look at the diagnostics and general graphics to see how things can change depending on sample size and repeated sampling.

The following code provides simulations of various time series models discussed in class. Let's examine the AR(1) model first to get familiar with some of the R syntax.

The following code provides a stationary AR(1) model with a lag 1 autocorrelation of 0.8. To help visualize the diagnostics more cleanly, I have simulated 10,000 observations. There are three plots. The first is time series itself which is hard to make anything form it since there are so many data points. The remaining two are the ACF and PACT respectively.

AR1 Behavior

```
AR1 <- arima.sim(list(ar = c(0.8)), 10000) # AR1 is just a vector of values. They wil
l be centered around 0 unless we add a shift or include some other source of variatio
n like a predictor.
par(mfrow = c(1, 3))
plot(1:10000, AR1, type = "l")
acf(AR1, main = "ACF")
pacf(AR1, main = "PACF")
```



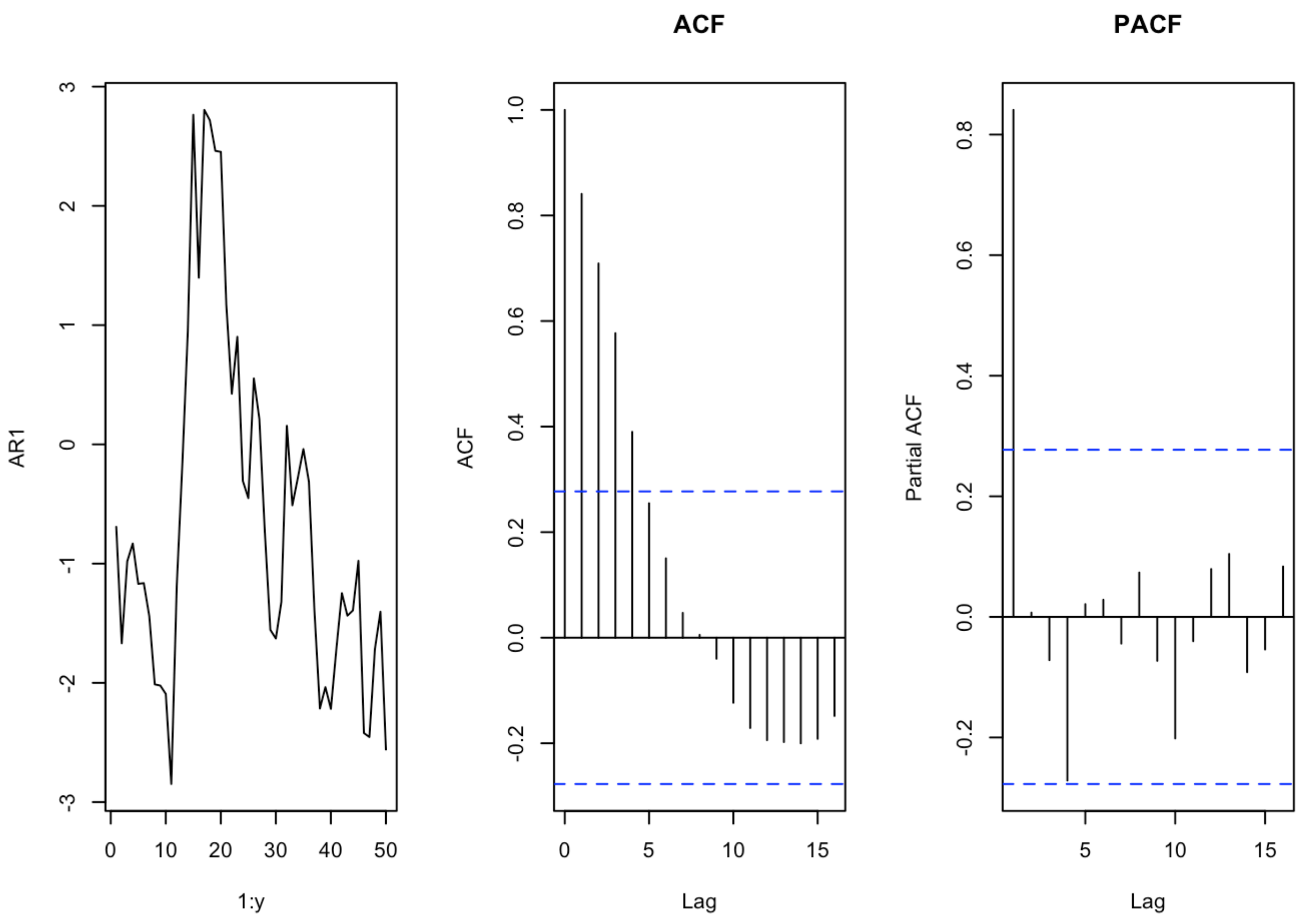
Homework Question

6. Using the rules of thumb table provided in live session, verify that the simulated data does in fact exhibit the same behaviors in the ACF and PACF plots that an AR(1) model would have. Verify also that the lag1 autocorrelation value is roughly 0.8 as expected.

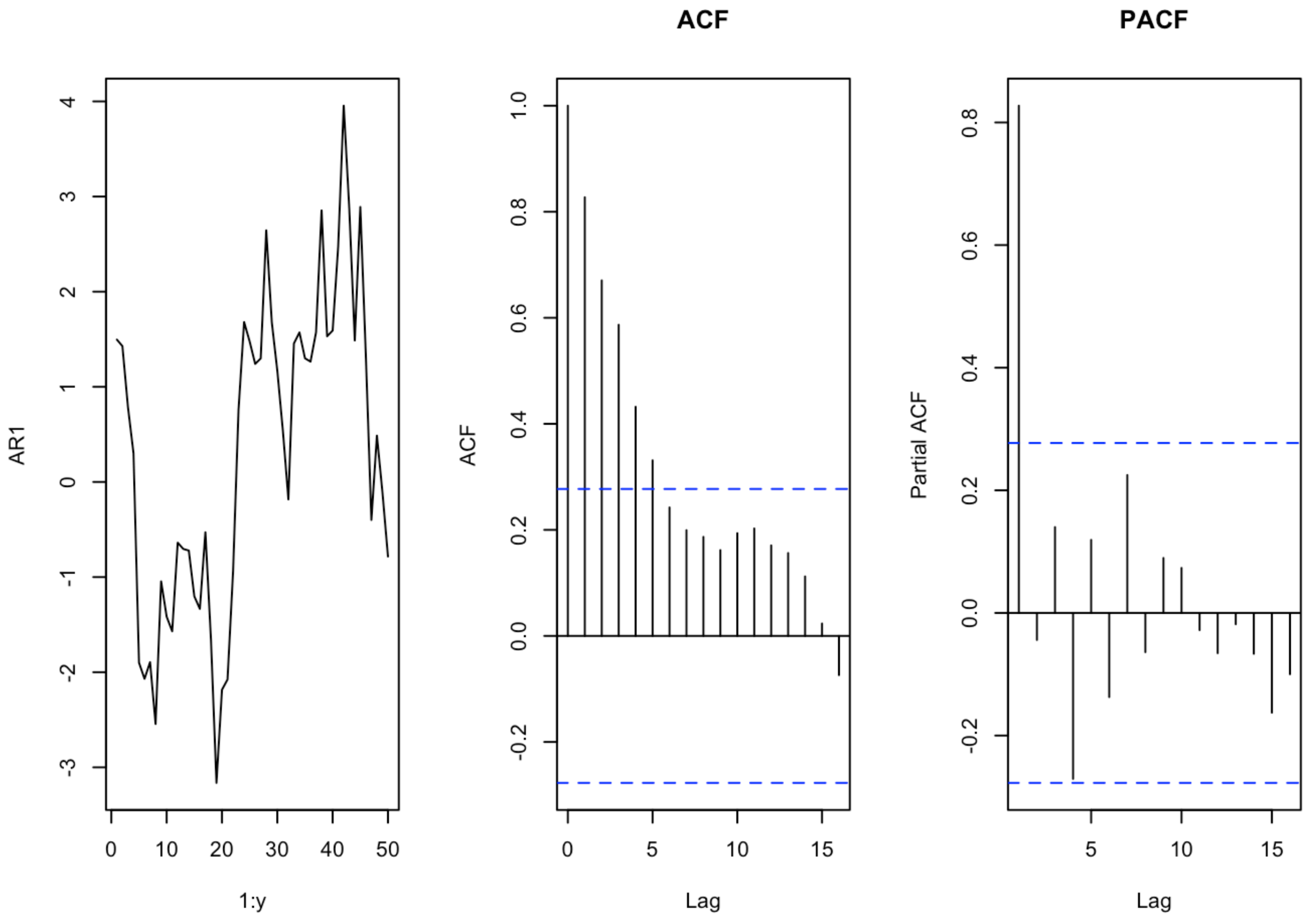
The ACF and PACF plots do indeed appear to exhibit the same behaviors that an AR(1) model should have, namely the ACF plot showing a general downward trend and the PACT plot showing an immediate drop off. The lag(1) autocorrelation is also roughly 0.8.

7. Repeat the previous code but only simulate a data set that has 50 observations rather than 10,000. Generate the same 3 plots that were generated before. Repeat this process 2 more times, each one having only 50 observations. The point of this exercise is to recognize in smaller data sets, the ACF and PACF plots are not perfect and gives the student some experience on the variation of the plots from data set to set (like examining qqplots for residual assumption checking in regression or ANOVA).

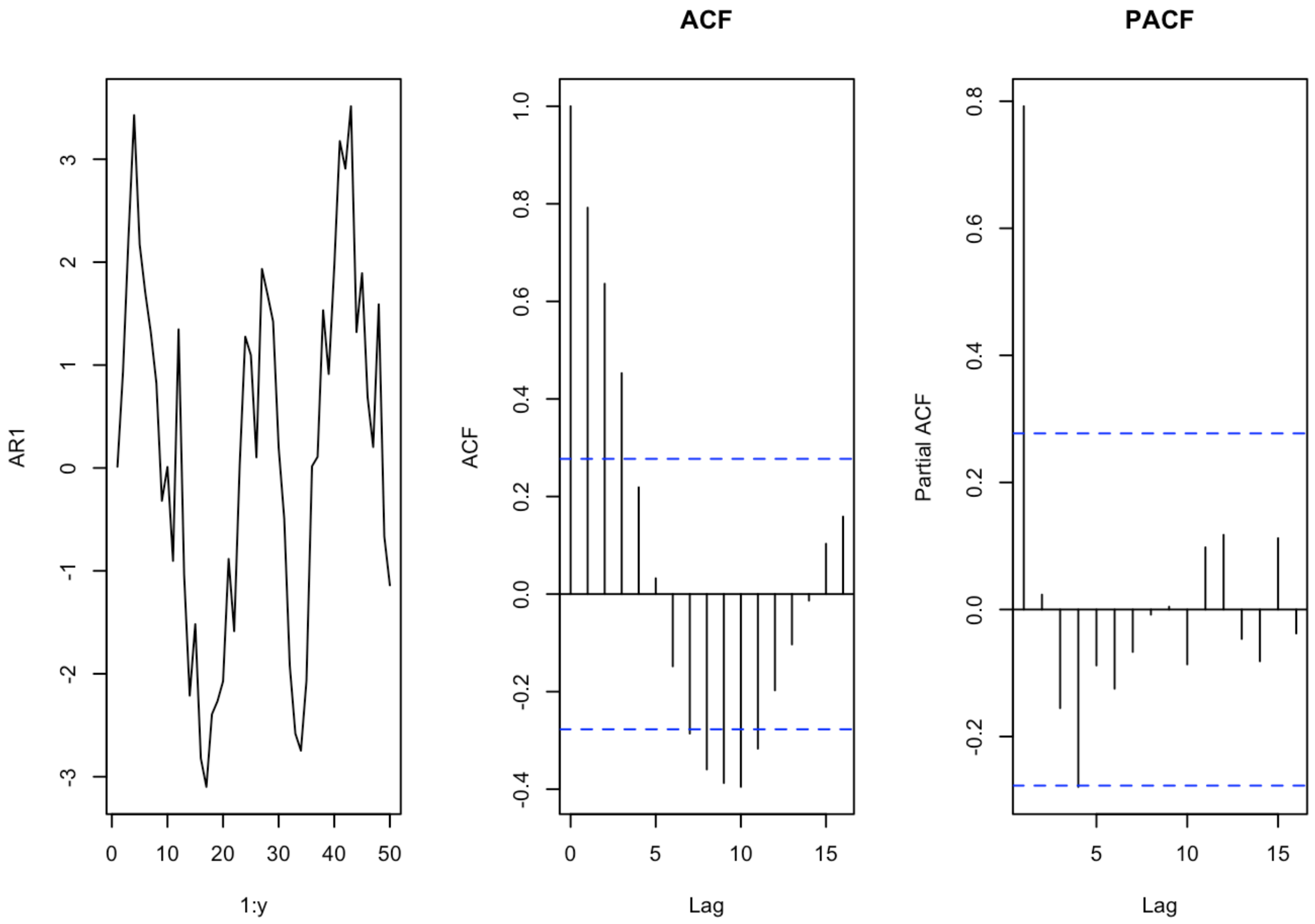
```r
sample_ar1 <- function(x, y) {
  AR1 <- arima.sim(list(ar = c(x)), y)
  par(mfrow = c(1, 3))
  plot(1:y, AR1, type = "l")
  acf(AR1, main = "ACF")
  pacf(AR1, main = "PACF")
}

sample_ar1(0.8, 50)
```



```r
sample_ar1(0.8, 50)
```



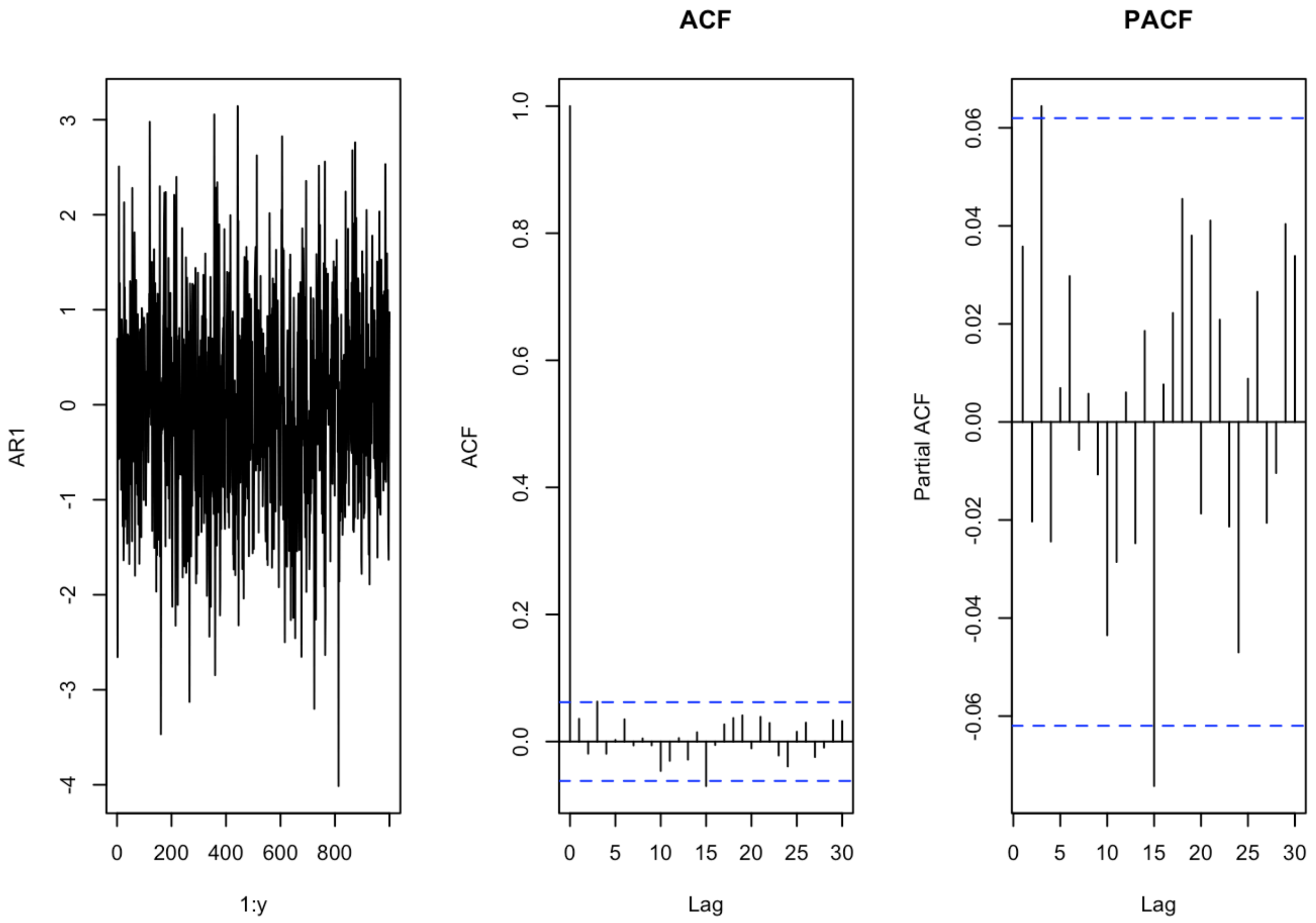
```
sample_ar1(0.8, 50)
```



8. If the lag 1 autocorrelation of a AR(1) model is set to 0, then there is no serial correlation present in the data. This is what we would like to see when we actually fit a time series model to the data, the resulting residuals from that model should behave uncorrelated (aka “white noise”). Simulate this senario using the above code and provide the ACF and PACF plots. This is what time series data (or residuals from a time series model) looks like when no serial correlation is present.

```
sample_ar1(0, 1000)
```

```
## Warning in min(Mod(polyroot(c(1, -model$ar)))): no non-missing arguments to min;
## returning Inf
```



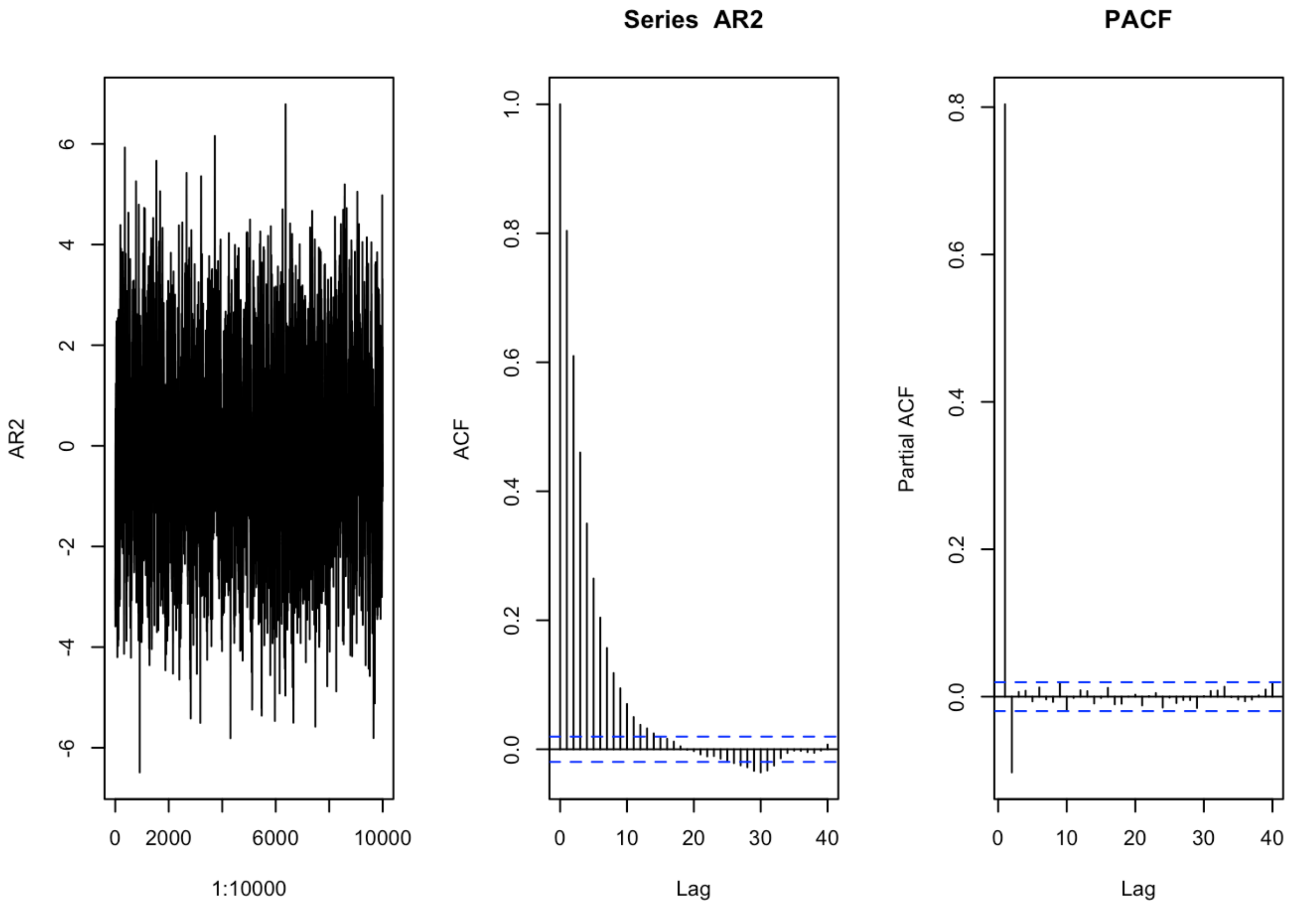
Exercise 2: More Time Series Model examples

Homework Questions

9. For each of the following scenarios, run the script and provide the 3 graphics. Verify that the ACF and PACF plots match the rules of thumb description for there specific scenario.

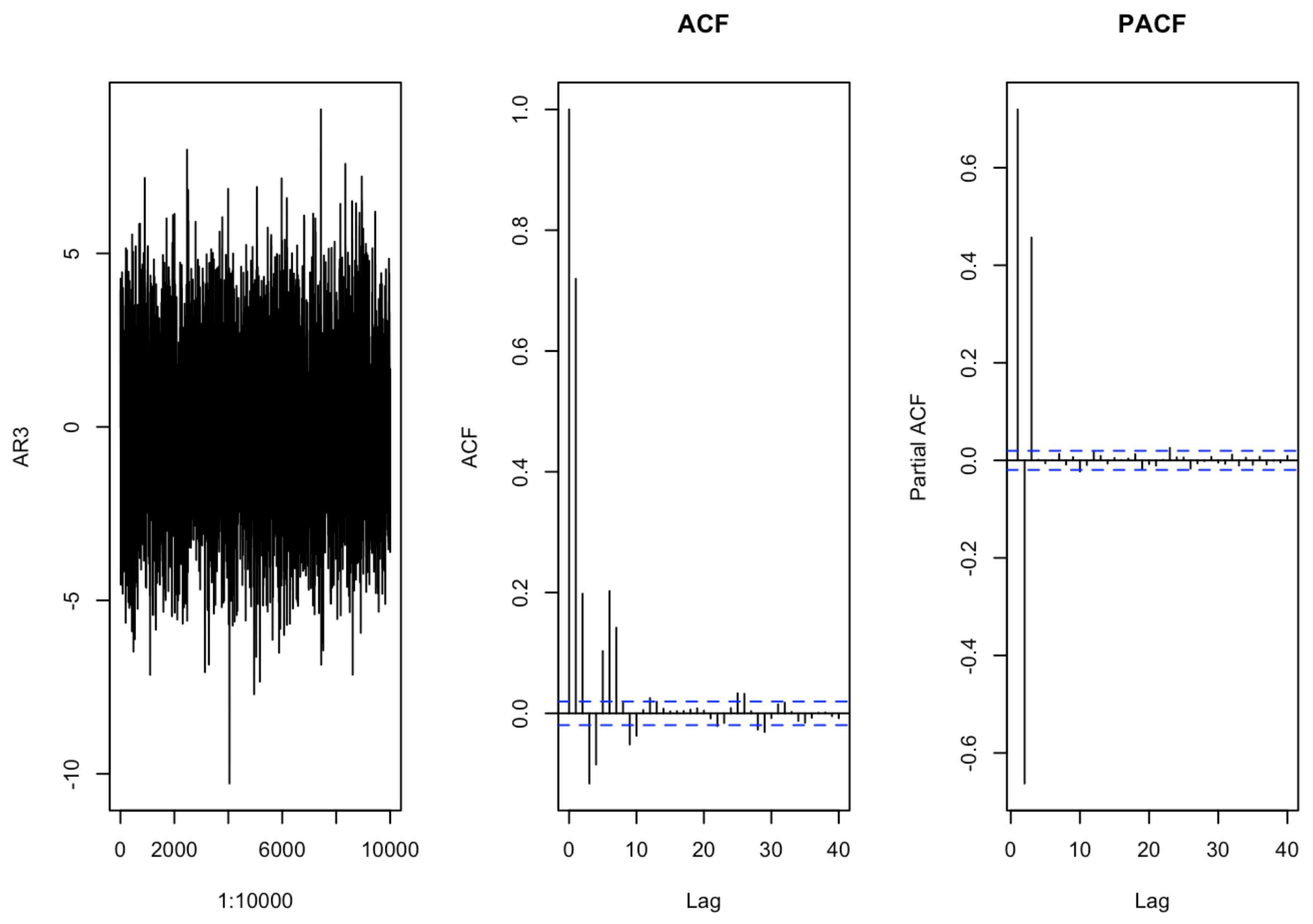
AR(2) Behavior

```
rho1 <- 0.8
rho2 <- 0.6
a1 <- (rho1 * (1 - rho2) / (1 - rho1^2))
a2 <- (rho2 - rho1^2) / (1 - rho1^2)
AR2 <- arima.sim(list(ar = c(a1, a2)), 10000)
par(mfrow = c(1, 3))
plot(1:10000, AR2, type = "l")
acf(AR2)
pacf(AR2, main = "PACF")
```



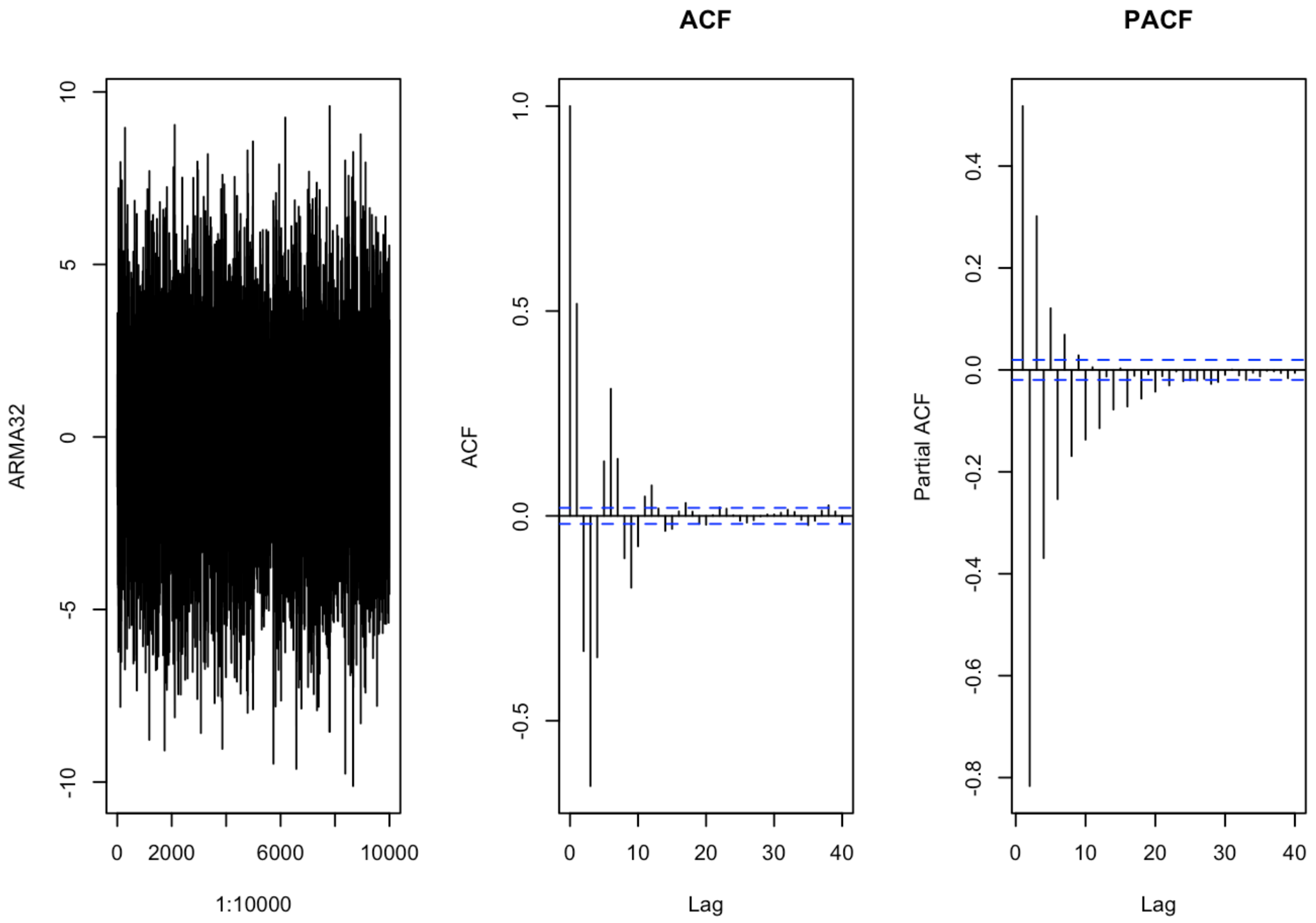
AR(3) Behavior

```
a1 <- 1.5
a2 <- -1.21
a3 <- .46
AR3 <- arima.sim(list(ar = c(a1, a2, a3)), 10000)
par(mfrow = c(1, 3))
plot(1:10000, AR3, type = "l")
acf(AR3, main = "ACF")
pacf(AR3, main = "PACF")
```



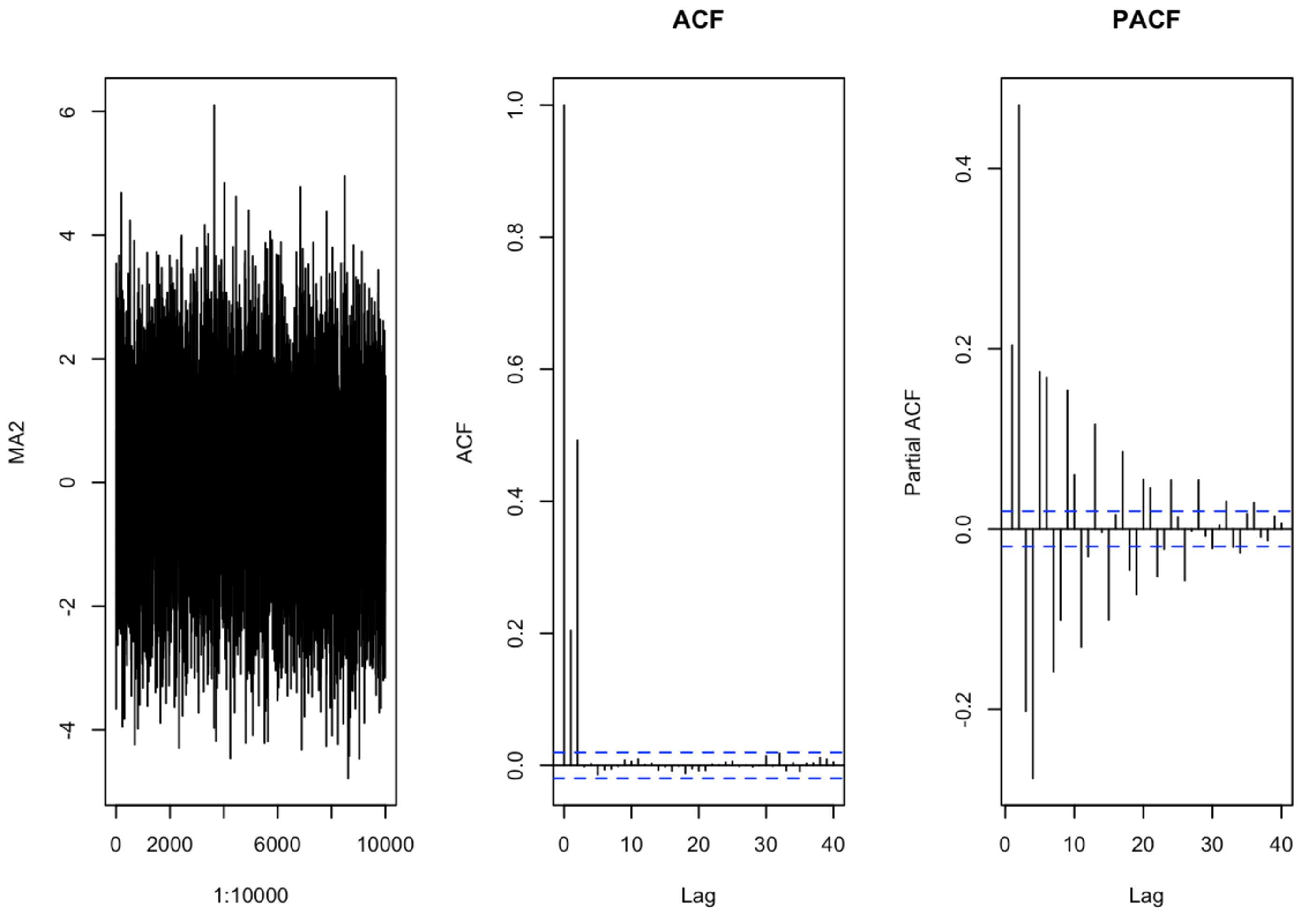
ARIMA(3, 2) Behavior

```
a1 <- 1.5
a2 <- -1.21
a3 <- .46
b1 <- -.2
b2 <- -.9
ARMA32 <- arima.sim(list(ar = c(a1, a2, a3), ma = c(b1, b2)), 10000)
par(mfrow = c(1, 3))
plot(1:10000, ARMA32, type = "l")
acf(ARMA32, main = "ACF")
pacf(ARMA32, main = "PACF")
```



Moving Average MA(2) Behavior

```
b1 <- .2
b2 <- .9
MA2 <- arima.sim(list(ma = c(b1, b2)), 10000)
par(mfrow = c(1, 3))
plot(1:10000, MA2, type = "l")
acf(MA2, main = "ACF")
pacf(MA2, main = "PACF")
```



Exercise 3: A good, simple example of time series

One of your fellow MSDS students was curious if he could model the fluctuations he sees in his electric bill. The data is included in Unit 4 zipped folder with the title “ElectricBill.csv”. In addition to just the amount due each month, he thought it might be helpful to include a predictor that could help aid in prediction. The potential predictor was the average monthly temperature of the city he lived in (Fort Worth).

Let’s start off by assuming that we did not have access to the temperature data (the student had to work to get that) and just work with what was the easiest available, the monthly electric bill. A plot of the time series is a good starting point.

```
bills <- read_csv(here::here("6372", "week 04", "ElectricBill.csv"))
```

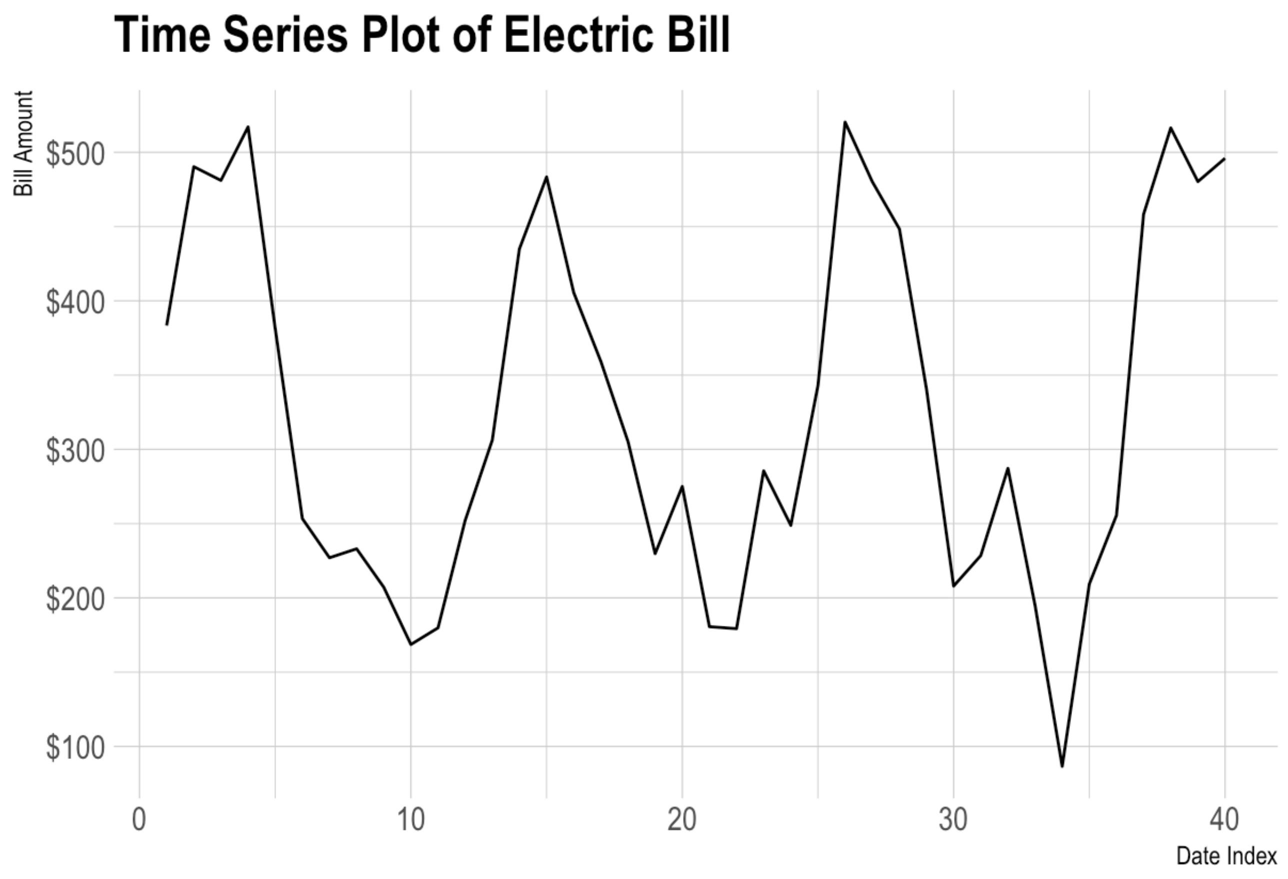
```
## Parsed with column specification:
## cols(
##   Date = col_character(),
##   Bill = col_double(),
##   AvgTemp = col_double()
## )
```

```
head(bills)
```

Date <chr>	Bill <dbl>	AvgTemp <dbl>
5/1/2015	383.37	71.40
6/1/2015	490.36	78.60
7/1/2015	481.11	80.00
8/1/2015	517.13	86.35
9/1/2015	381.30	79.15
10/1/2015	253.40	67.95

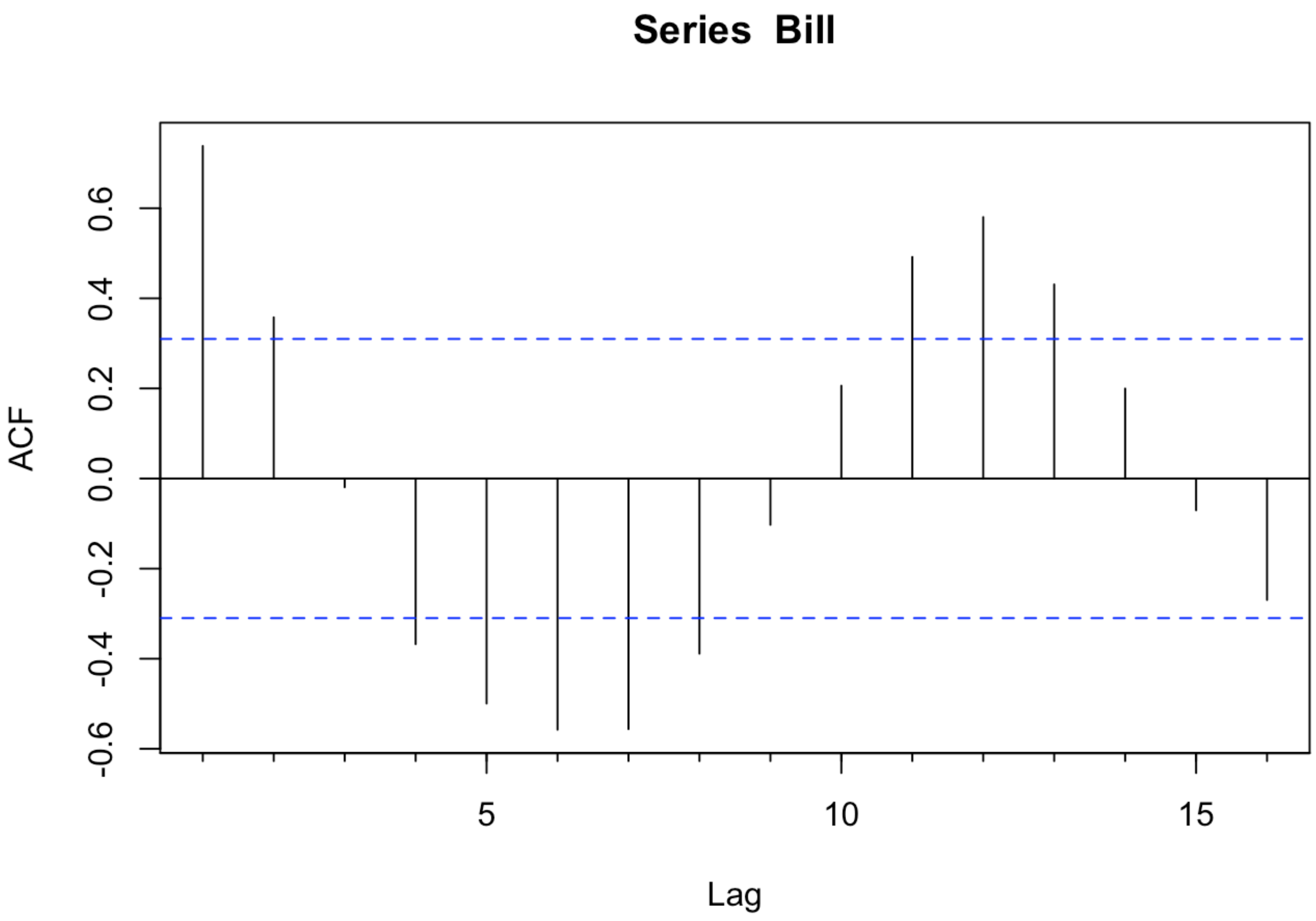
6 rows

```
bills$DateIndex <- 1:nrow(bills)

bills %>%
ggplot(aes(DateIndex, Bill)) +
  geom_line() +
  scale_y_continuous(labels = dollar) +
  labs(title = "Time Series Plot of Electric Bill",
       x = "Date Index",
       y = "Bill Amount") +
  theme_ipsum()
```

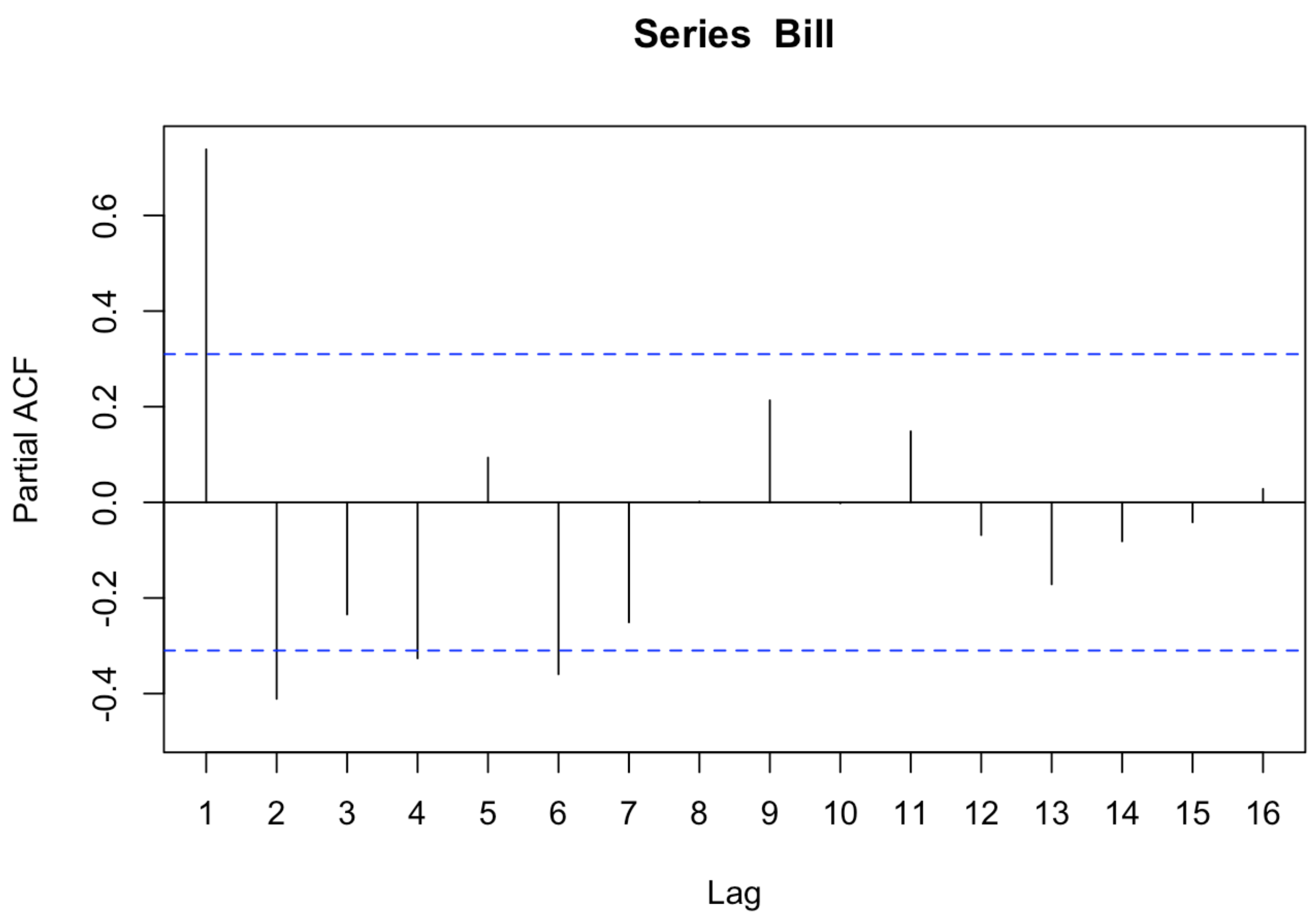


The plot of electric bill over time appears to show some cyclic behavior as expected. At face value, it appears that the time series has a common mean and variability appears to be roughly constant. The fact that high peaks tend to be pretty sharp while the valleys tend to be drawn out is a little bit of concern and could benefit from a log transformation. Before doing that, let's examine the ACF and PACF plots on the original data set.

```
attach(bills)
Acf(Bill)
```



```
Pacf(Bill)
```



From the ACF and PACF, we clearly see an evidence of serial correlation. There is some evidence of non-stationarity as the correlations from the larger lags are still pretty strong. However, with this small of a data set it is hard to tell. With a much larger data set, we could examine the lags farther out at least two lag 24 so we have to full years, to examine the cyclical behavior.

For completeness, I've included the Durbin Watson test statistic for auto correlation. The way this function works is that you have to feed the function a linear regression model. In this particular case, we are assuming the data is stationary already, with no need of any predictors so we can provide a regression model with just an intercept (make note that the an intercept model is just a means model). As expected, the Durbin Watson test rejects the null hypothesis, the time series has no serial correlation for a specific lag.

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
durbinWatsonTest(lm(Bill ~ 1), max.lag = 4)
```

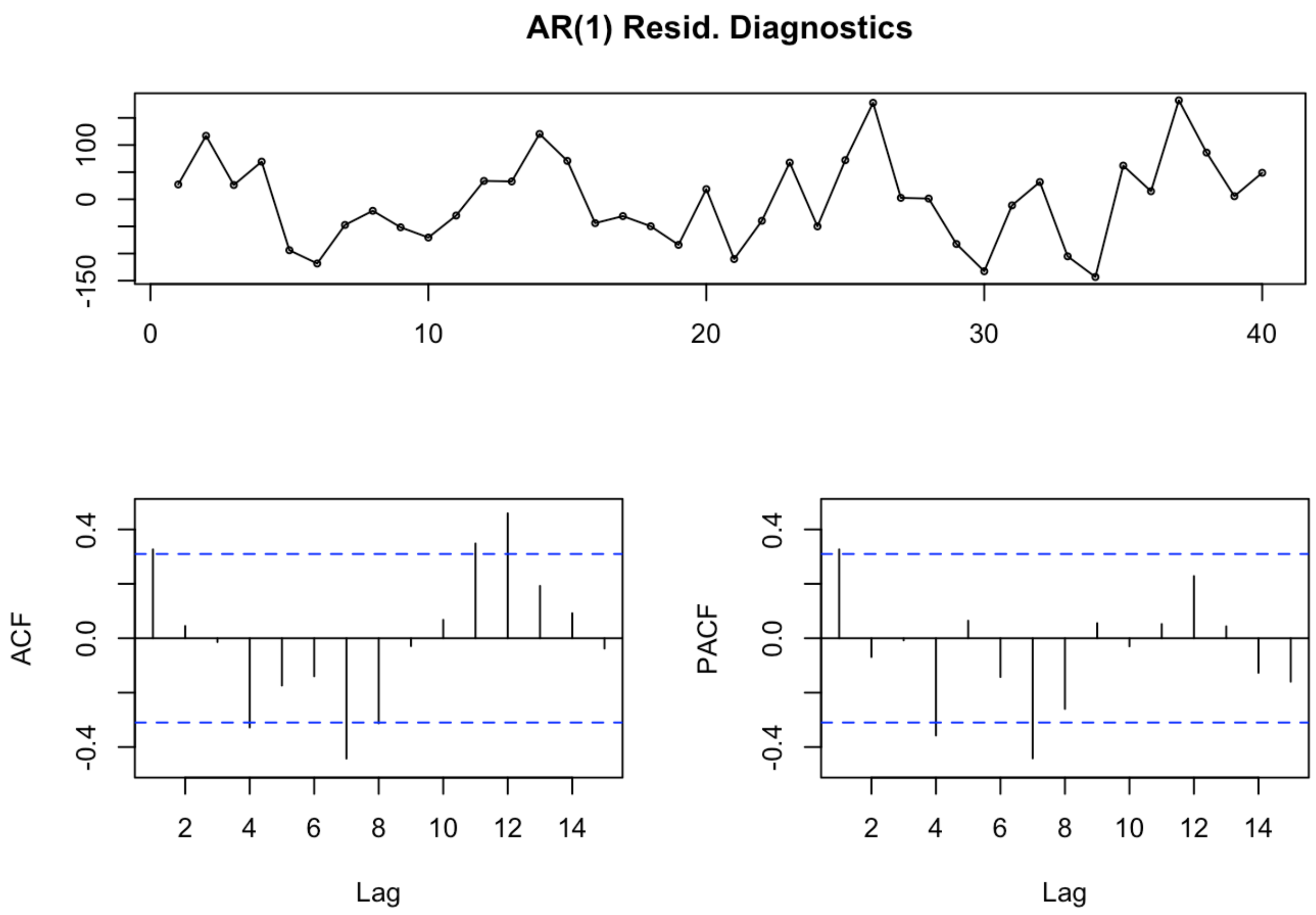
```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.7380876      0.469104   0.000
##    2       0.3577343      1.143470   0.004
##    3      -0.0194295      1.795373   0.718
##    4      -0.3676480      2.400021   0.072
##  Alternative hypothesis: rho[lag] != 0
```

With no major concerns that the time series is not stationary, we will begin to try to model the serial correlation that is present. Using our rules of thumb, it appears that an auto regressive model using 5 or 6 lags would be one candidate starting model. If one views the PACF as dying out gradually over time, an ARMA type model may be more appropriate. Let's first start out by fitting a few AR(p) models.

```
AR1 <- arima(Bill, order = c(1, 0, 0))
AR2 <- arima(Bill, order = c(2, 0, 0))
AR3 <- arima(Bill, order = c(3, 0, 0))
```

To examine the model fit, we can obtain residual diagnostics to see how well the autoregressive models have accounted for the serial correlation. If the model fits well, then the residuals should behave more and more like uncorrelated time series.

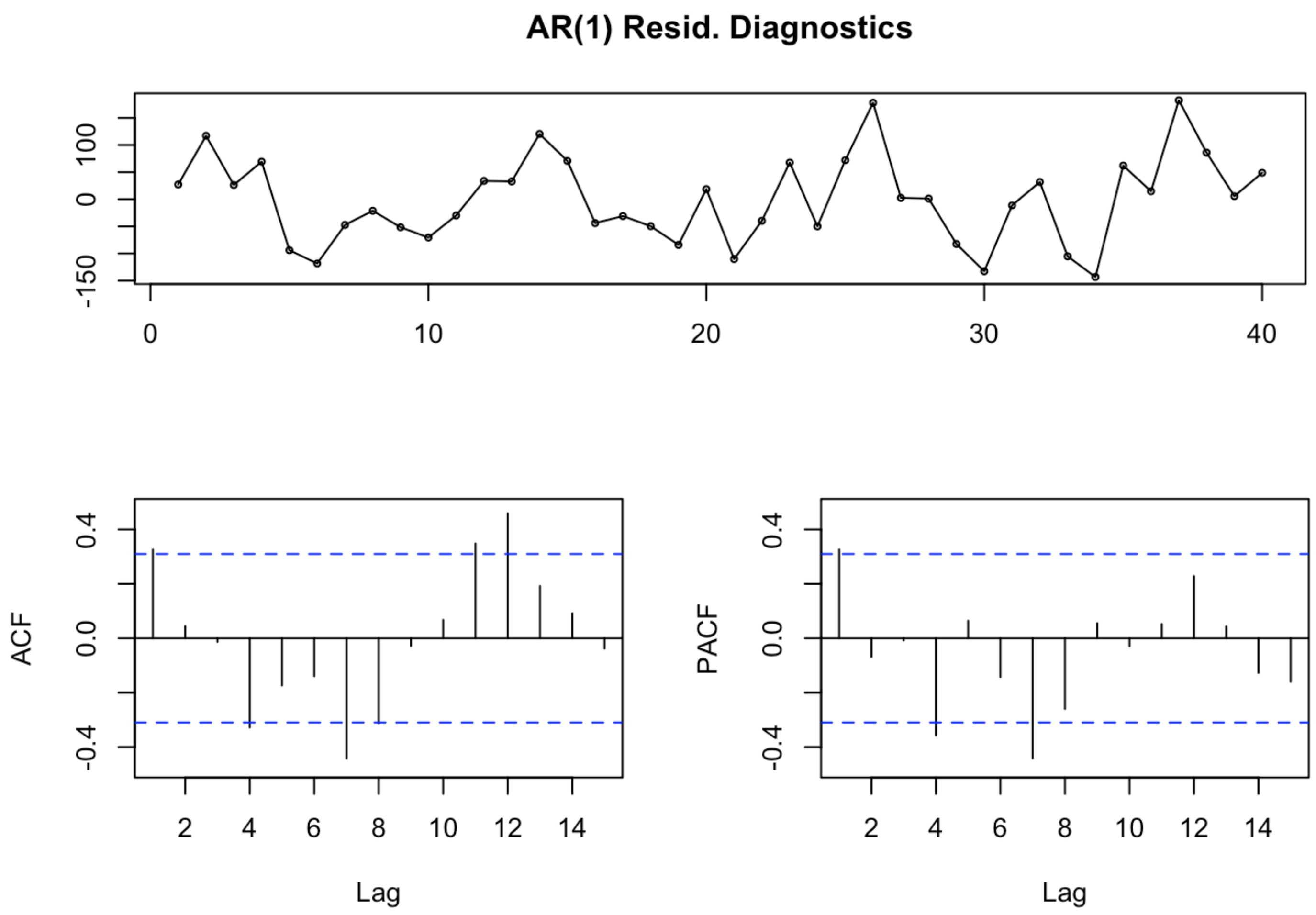
```
tsdisplay(residuals(AR1), lag.max = 15, main = "AR(1) Resid. Diagnostics")
```



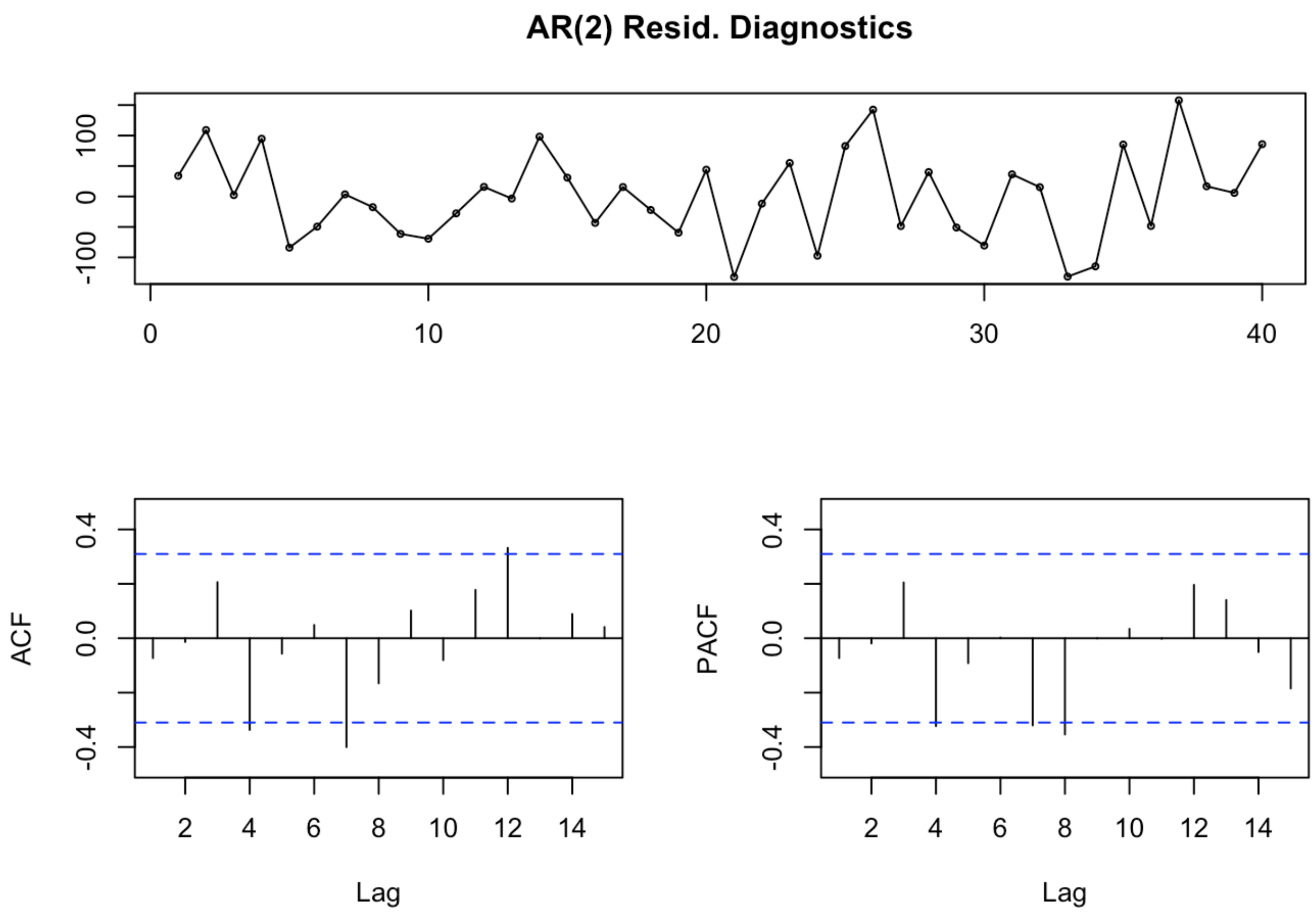
Homework Questions

10. The residuals from the AR(1) model above look to still have serial correlation present, so fit an additional AR(4) and AR(5) model. Compare the residual diagnostics of all 5 models to see if the ACF and PACF start to behave more like uncorrelated time series.

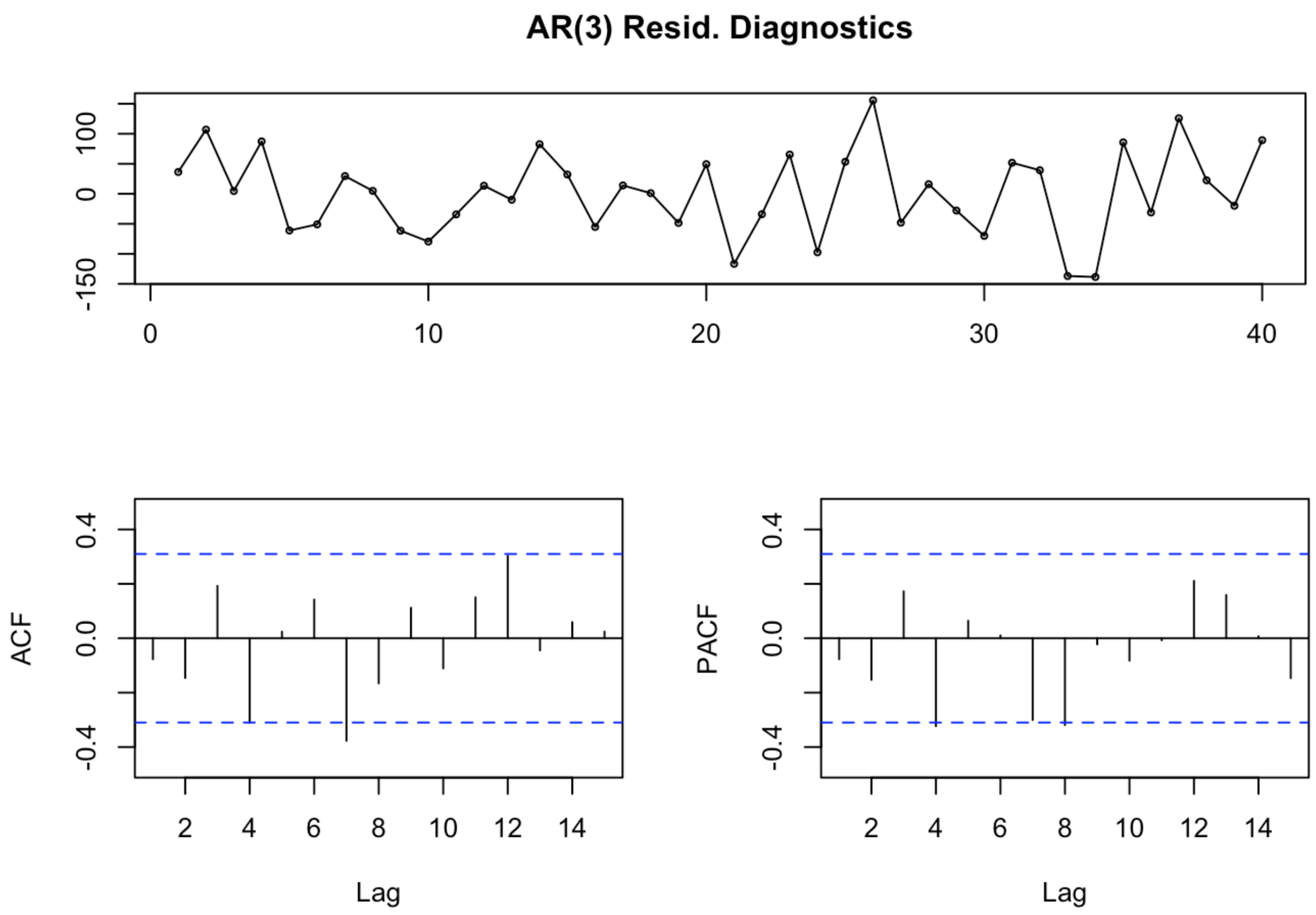
```
AR1 <- arima(Bill, order = c(1, 0, 0))
AR2 <- arima(Bill, order = c(2, 0, 0))
AR3 <- arima(Bill, order = c(3, 0, 0))
AR4 <- arima(Bill, order = c(4, 0, 0))
AR5 <- arima(Bill, order = c(5, 0, 0))

tsdisplay(residuals(AR1), lag.max = 15, main = "AR(1) Resid. Diagnostics")
```



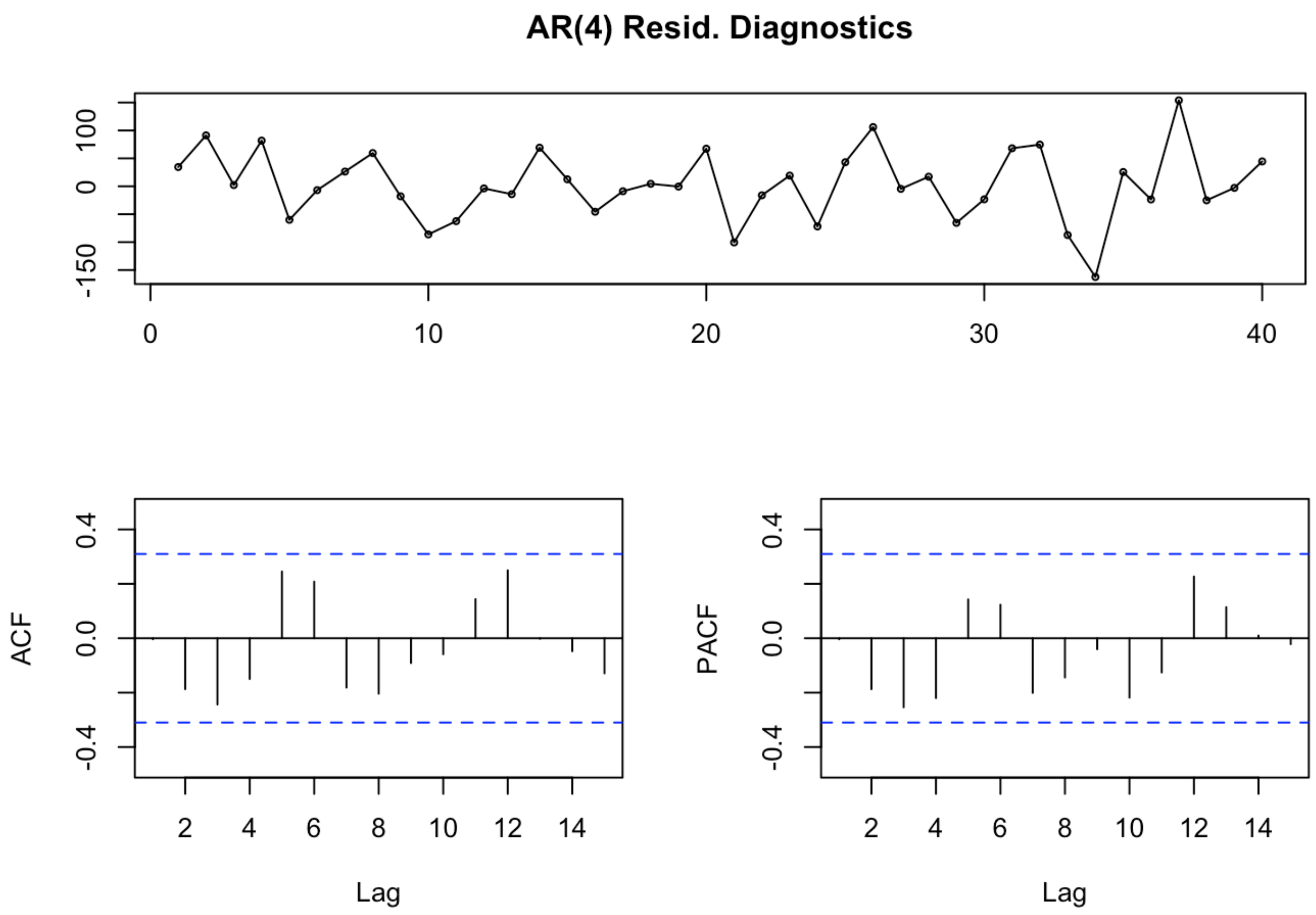
```
tsdisplay(residuals(AR2), lag.max = 15, main = "AR(2) Resid. Diagnostics")
```



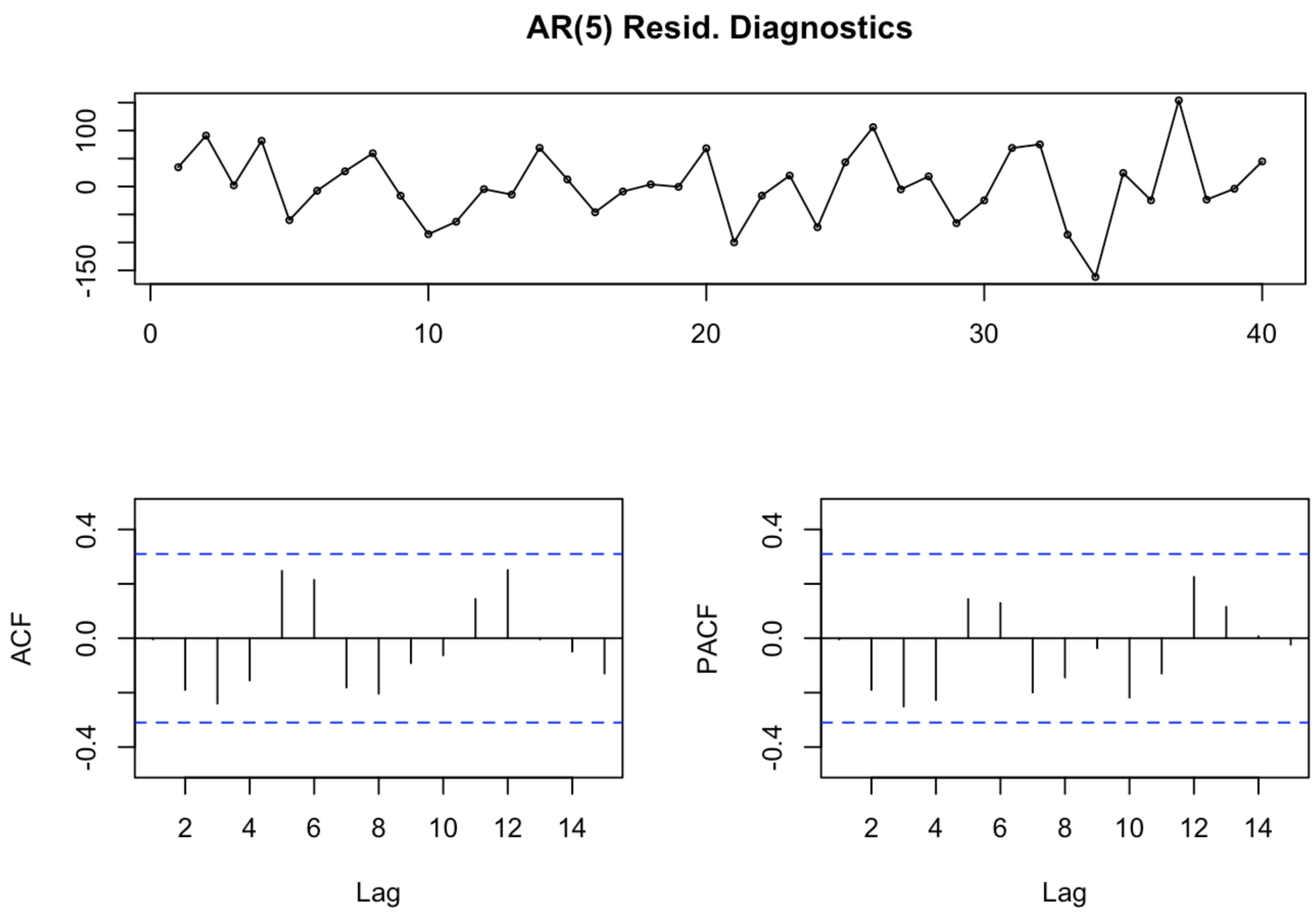
```
tsdisplay(residuals(AR3), lag.max = 15, main = "AR(3) Resid. Diagnostics")
```



```
tsdisplay(residuals(AR4), lag.max = 15, main = "AR(4) Resid. Diagnostics")
```



```
tsdisplay(residuals(AR5), lag.max = 15, main = "AR(5) Resid. Diagnostics")
```



11. Use the AIC function (ex: AIC(AR1)) to obtain the AIC of each of the 5 models from #2. Does the model that yields the lowest AIC provide a residual plot that looks the best in terms of having removed the serial correlation?

```
bills_aic <- tibble(ar1 = AIC(AR1),
                    ar2 = AIC(AR2),
                    ar3 = AIC(AR3),
                    ar4 = AIC(AR4),
                    ar5 = AIC(AR5))

bills_aic %>%
  gt()
```

ar1	ar2	ar3	ar4	ar5
470.0568	464.3603	464.4424	458.2644	460.2588

AR(4) has the lowest AIC value (458.2644), but AR(5) is close behind at 460.2588. Both AR(4) and AR(5) appear to be very similar on the ACF and PACF plots.

Rather than iteratively fitting models and checking AIC metrics, we can perform an automated procedure that searches a more diverse set of ARIMA models. Note here that this algorithm by default uses a stepwise procedure and doesn't necessary find a global minimum AIC value. For example, let's run the automated procedure using all of the defaults.

```
ARIMA.fit <- auto.arima(Bill, seasonal = FALSE)
ARIMA.fit
```

```
## Series: Bill
## ARIMA(2,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2      mean
##       1.0811  -0.4344  326.1589
## s.e.  0.1419   0.1472   31.8184
##
## sigma^2 estimated as 5528:  log likelihood=-228.18
## AIC=464.36   AICc=465.5   BIC=471.12
```

Notice here the automated procedure selected the AR(2) model which we know from the previous exercises that AR(4) has a lower AIC. This is due to the stepwise selection process of auto.arima. To get them to correspond,

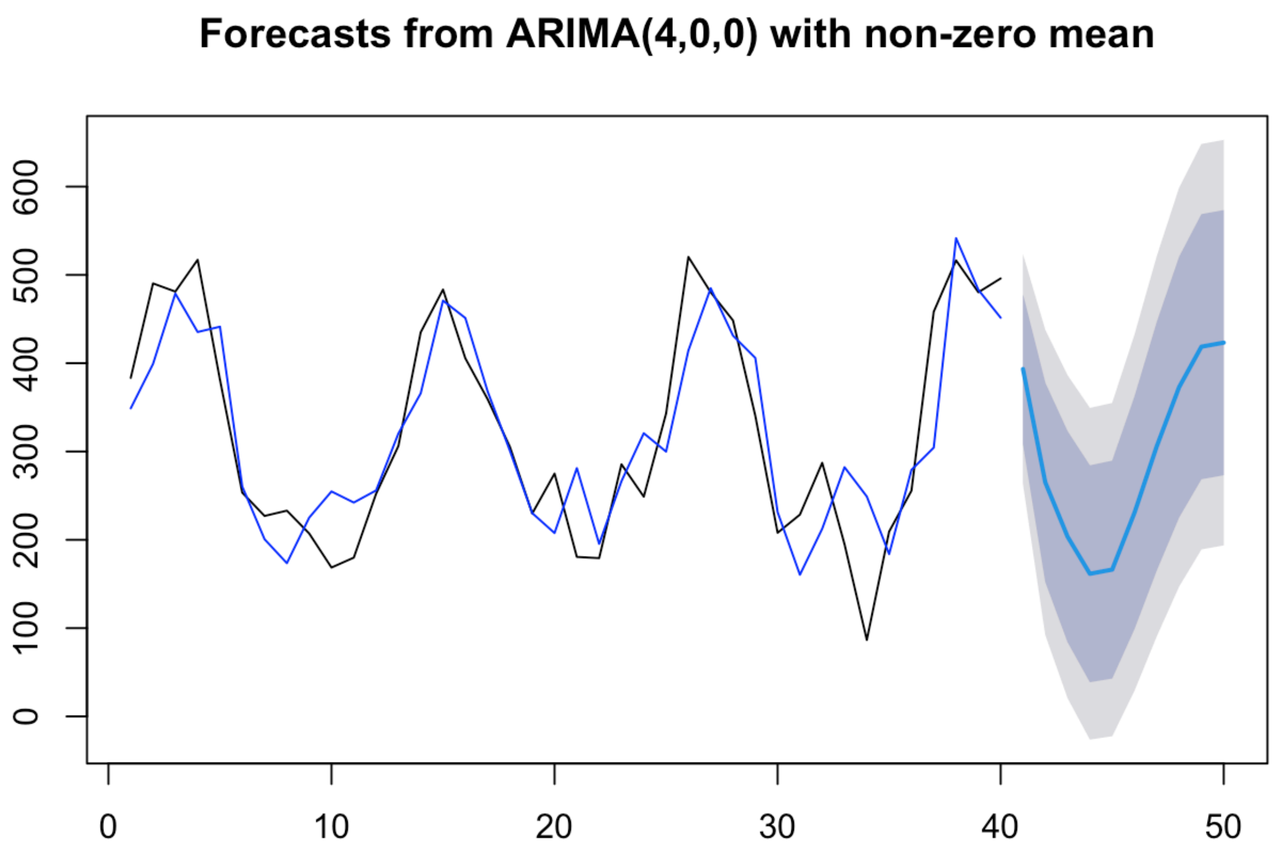
```
ARIMA.fit <- auto.arima(Bill, seasonal = FALSE, stepwise = FALSE)
ARIMA.fit
```

```
## Series: Bill
## ARIMA(4,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ar3      ar4      mean
##       0.8751  -0.3024  0.1918  -0.4628  311.7293
## s.e.  0.1415   0.2025  0.1998   0.1473   14.7687
##
## sigma^2 estimated as 4386:  log likelihood=-223.13
## AIC=458.26   AICc=460.81   BIC=468.4
```

This function has a ton of bells and whistles and can also handle seasonal ARIMA models. Take a look at `?auto.arima`.

Providing forecasts of a final model fit is relatively simple to do. Syntactically, the h in the forecast function provides the future forecast up to h time points ahead. In addition, to the forecasts, I've overlaid the model fit on the previous data to get look at that as well.

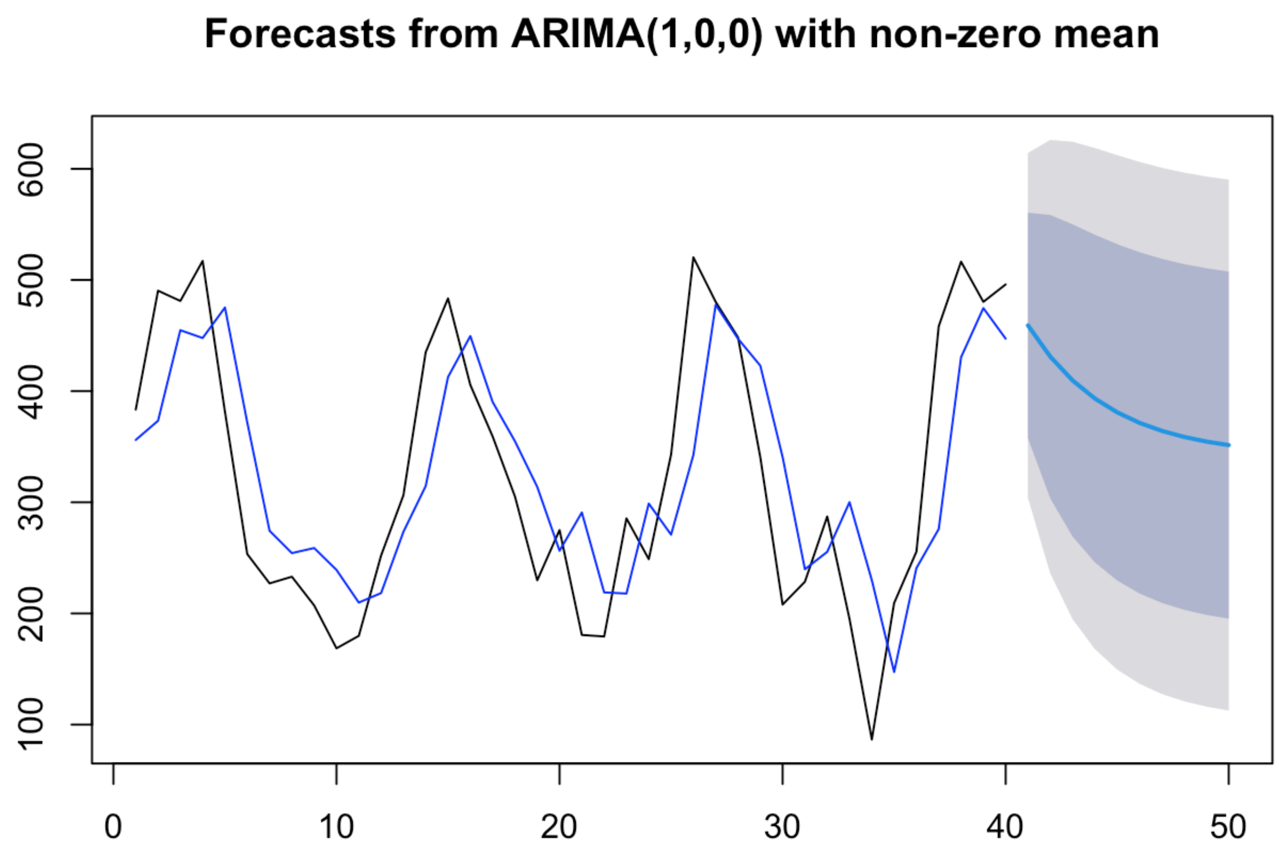
```
plot(forecast(ARIMA.fit, h = 10))
points(1:length(Bill), fitted(ARIMA.fit), type = "l", col = "blue")
```



Homework Exercises

12. We know from previous explorations that the AR(1) model fits poorly as serial correlation is still un accounted for. Provide a forecast from the AR(1) model and compare it to the previous AR(4). What properties do the forecasts from this predictive model (AR1) lack in comparison to the AR(4)? What about the prediction interval bands?

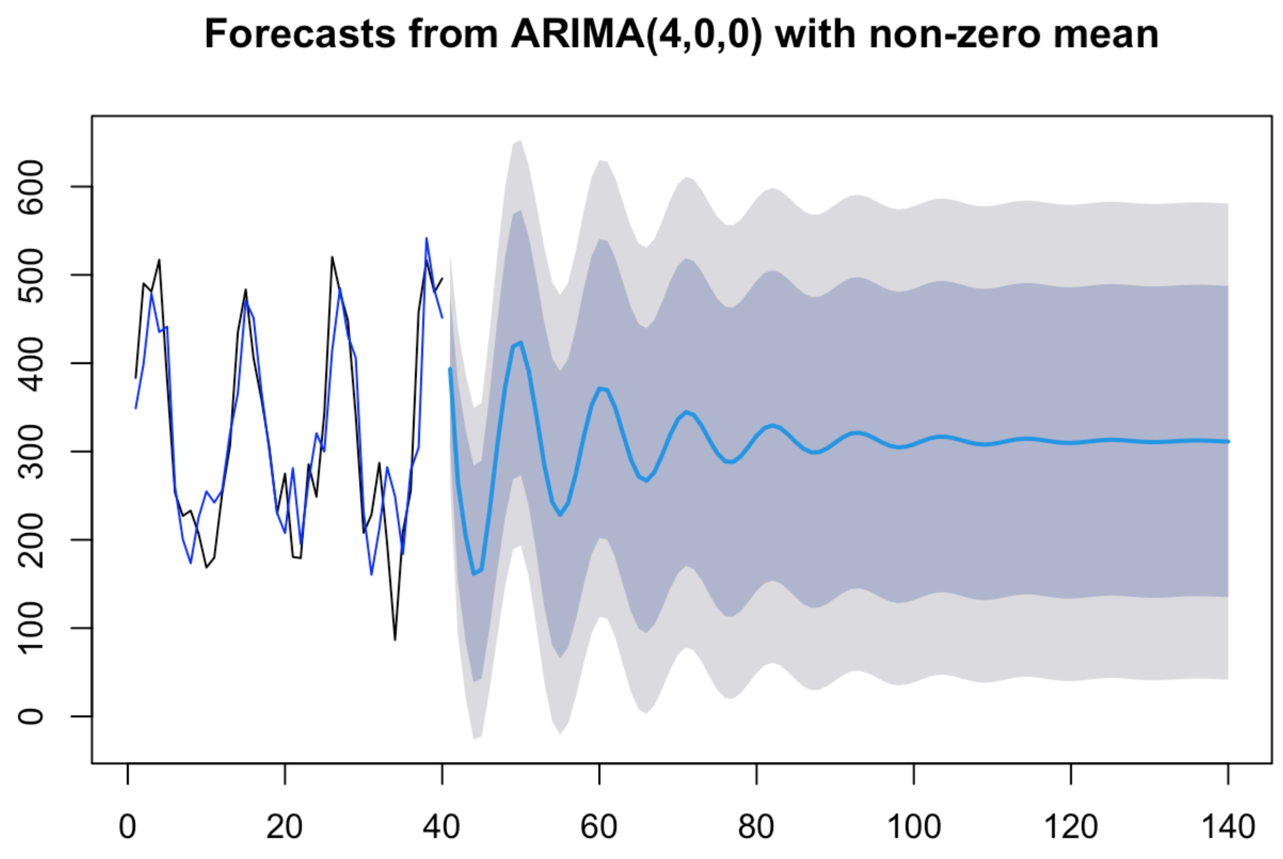
```
AR1 <- arima(Bill, order = c(1, 0, 0))
plot(forecast(AR1, h = 10))
points(1:length(Bill), fitted(AR1), type = "l", col = "blue")
```



The AR(1) model has much wider prediction interval bands and the forecast shows a downward slope because it is only looking at the point behind it. In comparison, the AR(4) model shows tighter prediction interval bands and the model understands that after declining, the values go back up.

13. Provide another plot of the forecasts for the AR(4) model but forecast out to the next 100 observations. Sometimes students are blown away and not satisfied by the result. Can you explain, in a common sense way, why the forecasts behave the way they do?

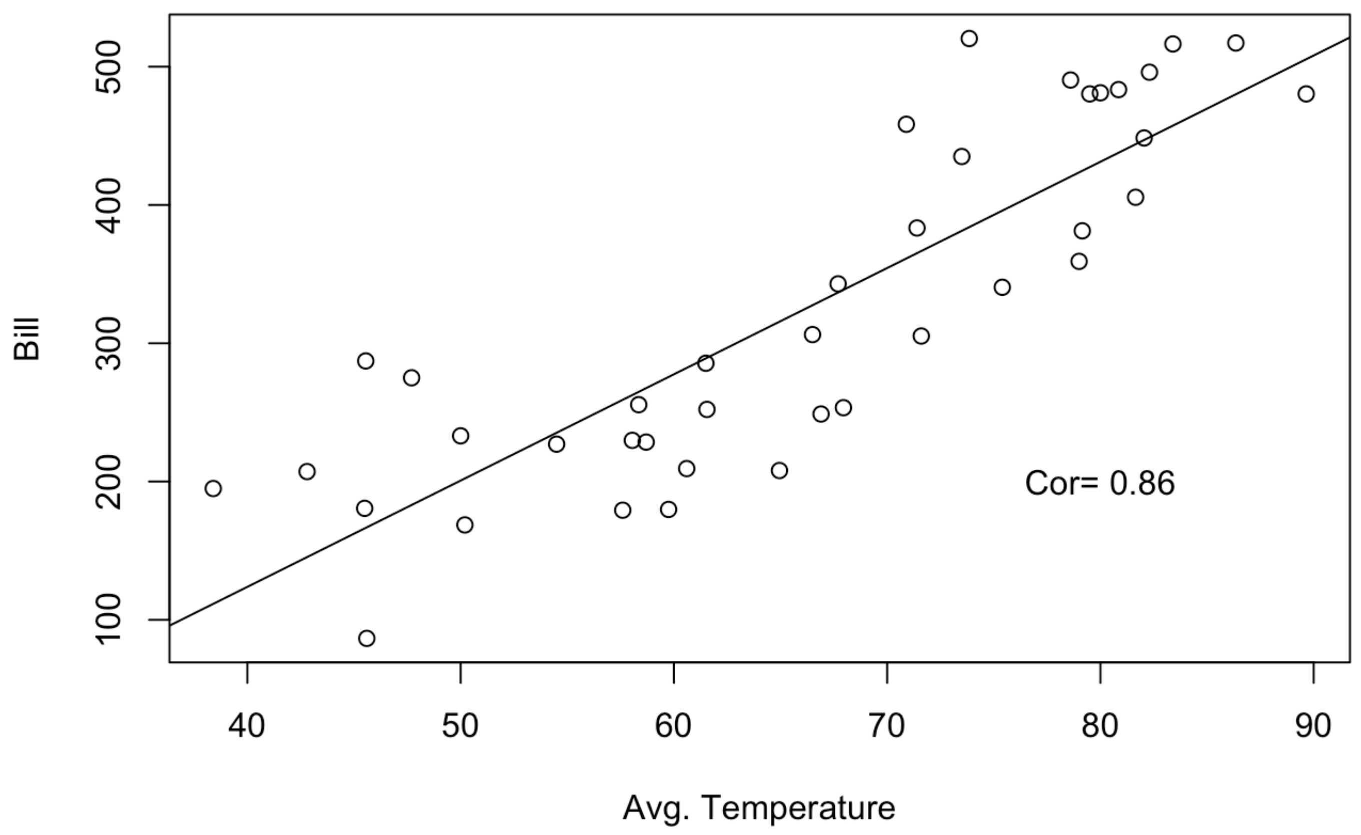
```
plot(forecast(ARIMA.fit, h = 100))
points(1:length(Bill), fitted(ARIMA.fit), type = "l", col = "blue")
```



As the forecast moves beyond known data, it will gradually drift towards regression because the model is forecasting on forecasted data.

The eclectic bill data set also contained some additional information, local area monthly temperatures. This could be helpful to include as a predictor if it can help explain the variation in the response. Like any regression plot, let's examine the a scatter plot of bills versus average temperature.

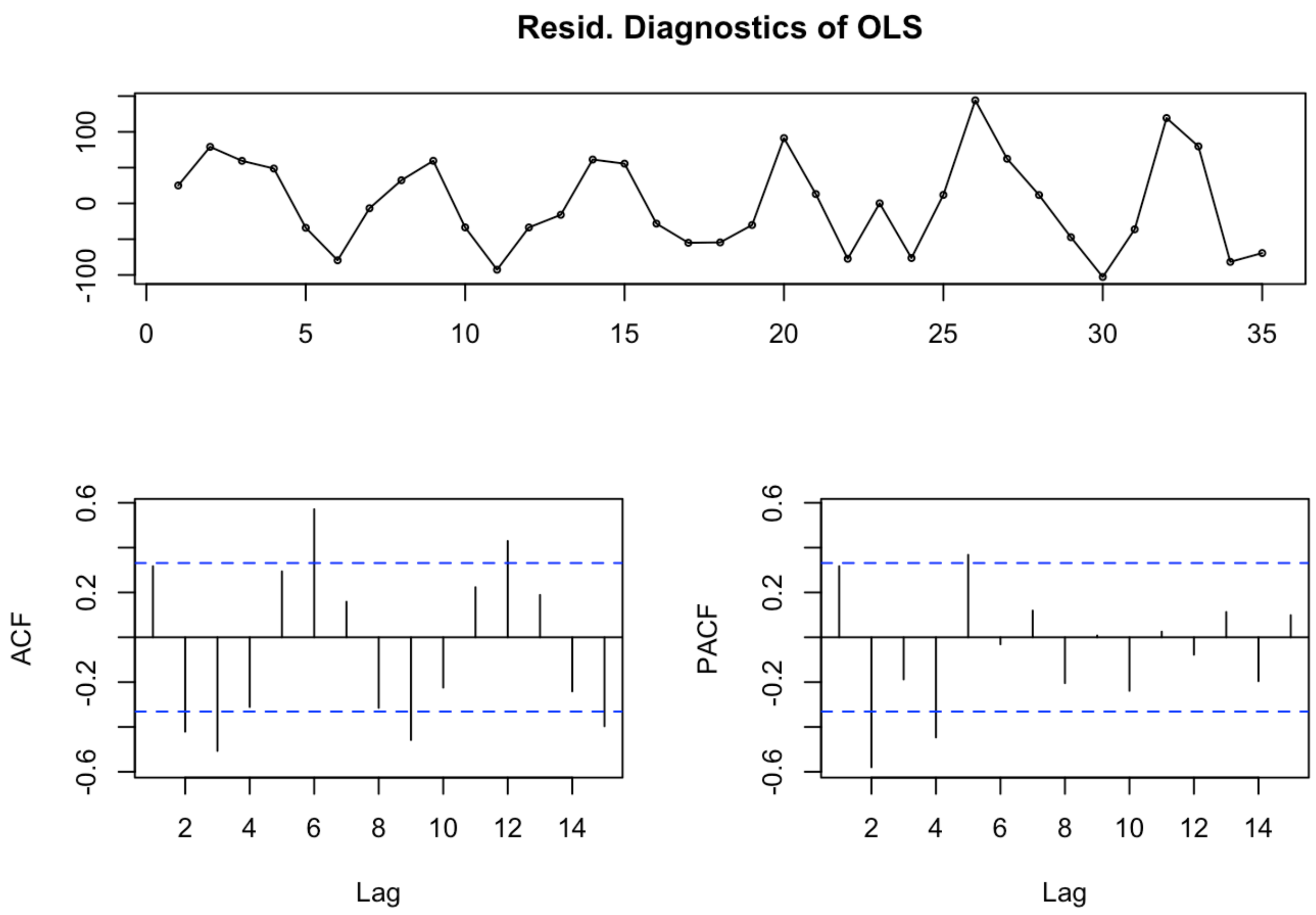
```
plot(AvgTemp, Bill, xlab = "Avg. Temperature")
ols <- lm(Bill ~ AvgTemp)
abline(ols)
text(80, 200, paste("Cor=", round(cor(Bill, AvgTemp), 2)))
```



The correlation is quite strong (maybe even a little quadratic) suggesting that this could be used to help improve our forecasts. Including a predictor is straightforward, but keep in mind by including a deterministic component in your model, you are making the decision that the original time series model is not stationary any more.

Also, logistically speaking, to make future predictions we now need to have access to the observed observations of the predictor. Having said this, moving forward we are going to hold out the last 5 observations of the original time series, to make forecasts since we need information on the temperatures. Let's examine the ACF and PACF of the residuals after regressing Bill on AvgTemp.

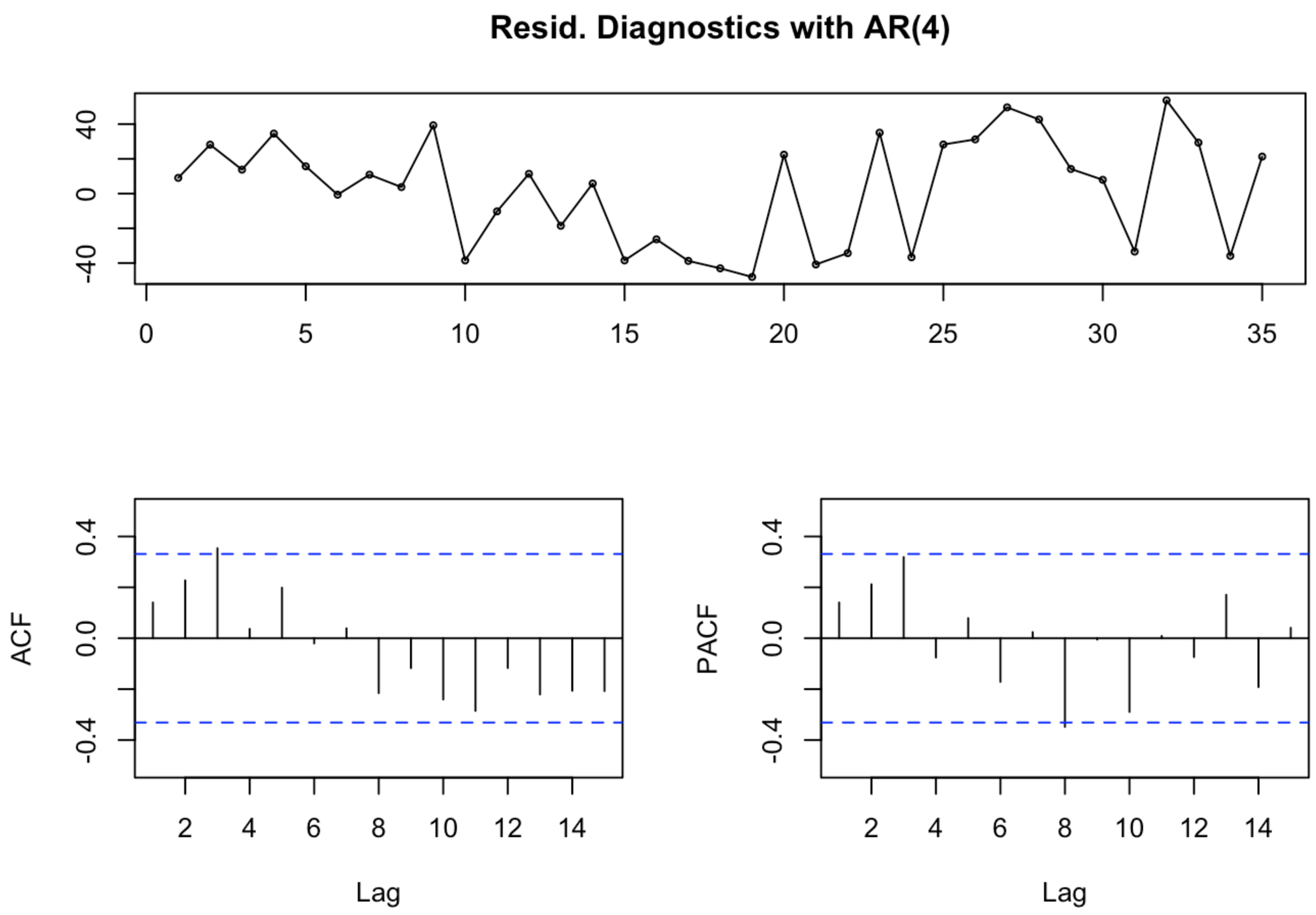
```
holdout.test <- window(ts(Bill), start = 36)
train <- Bill[1:35]
predictor <- AvgTemp[1:35]
simpleols <- arima(train, order = c(0, 0, 0), xreg = predictor)
tsdisplay(residuals(simpleols), lag.max = 15, main = "Resid. Diagnostics of OLS")
```



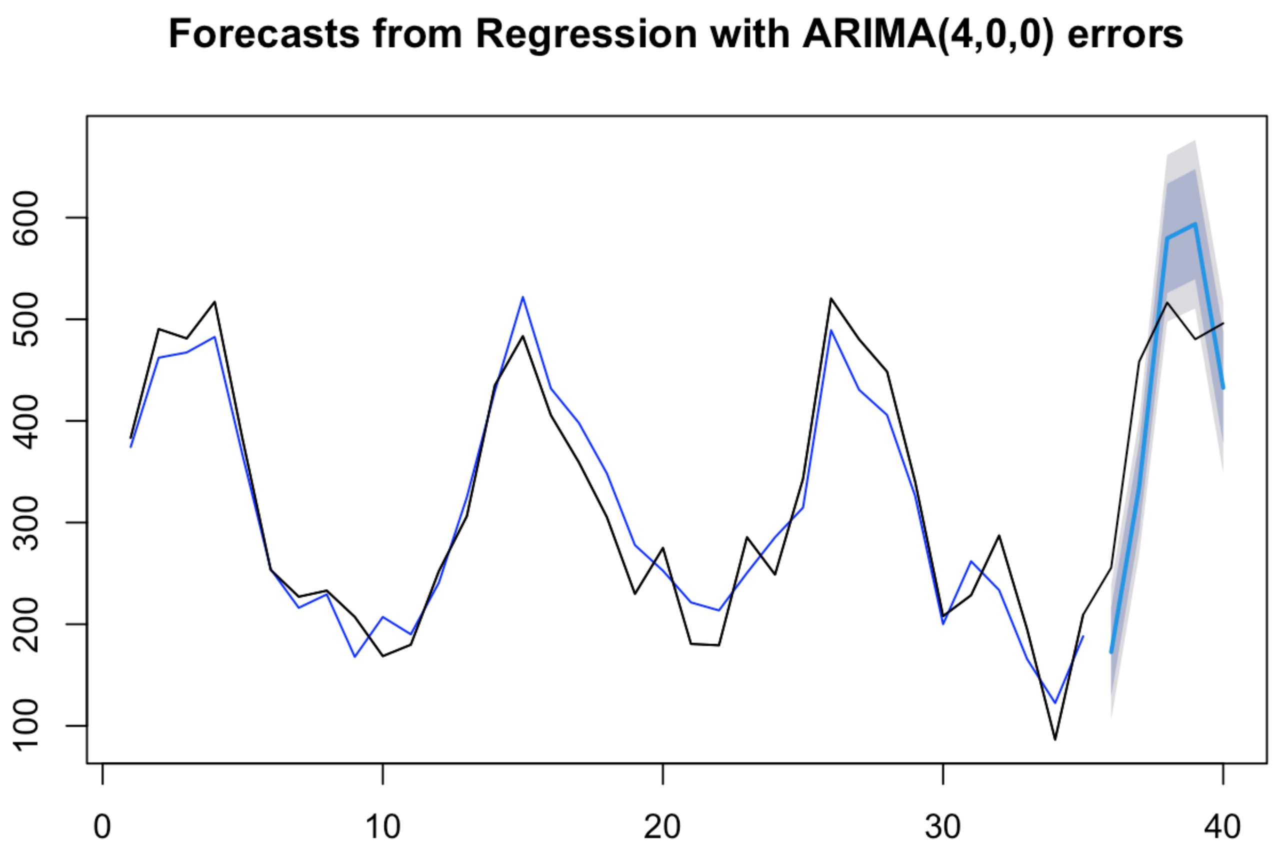
```
ARIMA.with.Pred <-
  auto.arima(train, xreg = predictor, stepwise = FALSE)

ARIMA.with.Pred
```

```
## Series: train
## Regression with ARIMA(4,0,0) errors
##
## Coefficients:
##           ar1      ar2      ar3      ar4  intercept    xreg
##       -0.0106  -0.7357  -0.1832  -0.7500  -235.4348  8.3784
## s.e.   0.1173   0.1139   0.1138   0.1095    16.2467  0.2481
##
## sigma^2 estimated as 1136:  log likelihood=-171.98
## AIC=357.97   AICc=362.11   BIC=368.85
```

```
tsdisplay(residuals(ARIMA.with.Pred),
          lag.max = 15,
          main = "Resid. Diagnostics with AR(4)")
```



```
# plot(forecast(
#   ARIMA.with.Pred,
#   h = 5,
#   xreg = data.frame(predictor = AvgTemp[36:40])
# ))

plot(forecast(
  ARIMA.with.Pred,
  h = 5,
  xreg = AvgTemp[36:40]
))

points(1:length(train),
       fitted(ARIMA.with.Pred),
       type = "l",
       col = "blue")

points(1:40, Bill, type = "l")
```



The most obvious benefit of including the predictor can be seen by the prediction interval band of the forecast. By modeling out variation in the data, their is less variability in the residuals which creates less uncertainty in the forecast. However, like any regression model, if the model developed using any set of predictors is not accurate (Bias/Variance trade off) then forecasts will suffer from the same issues.

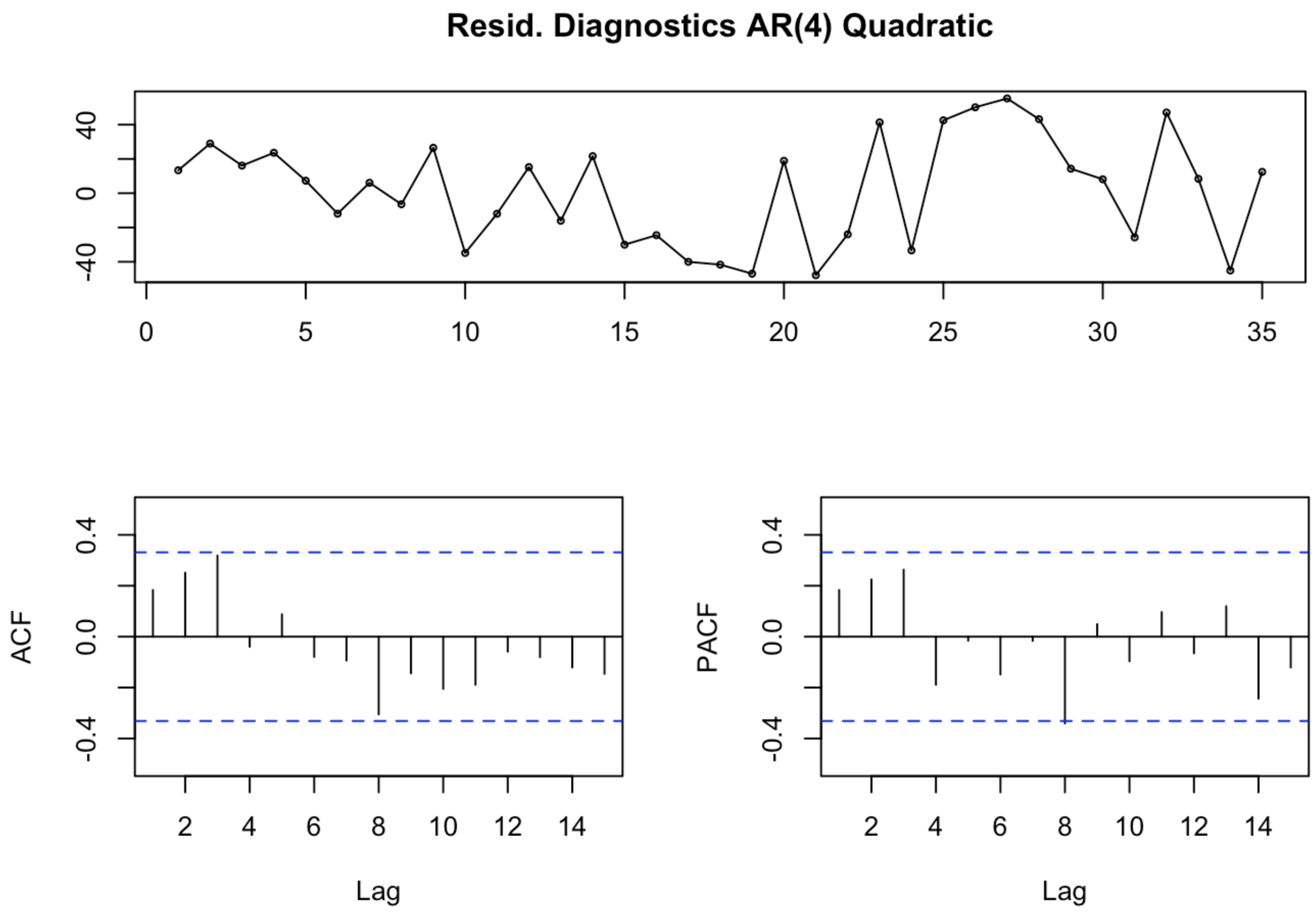
The natural followup question when incorporating predictors is deciding on the best fit and safe guarding against under and over fitting those predictors. To illustrate this I've included a second model fit including a quadratic term. Since we have test set (only 5 observations here but you get the point), we can produce ASE type metrics to compare models.

New Quadratic Fit

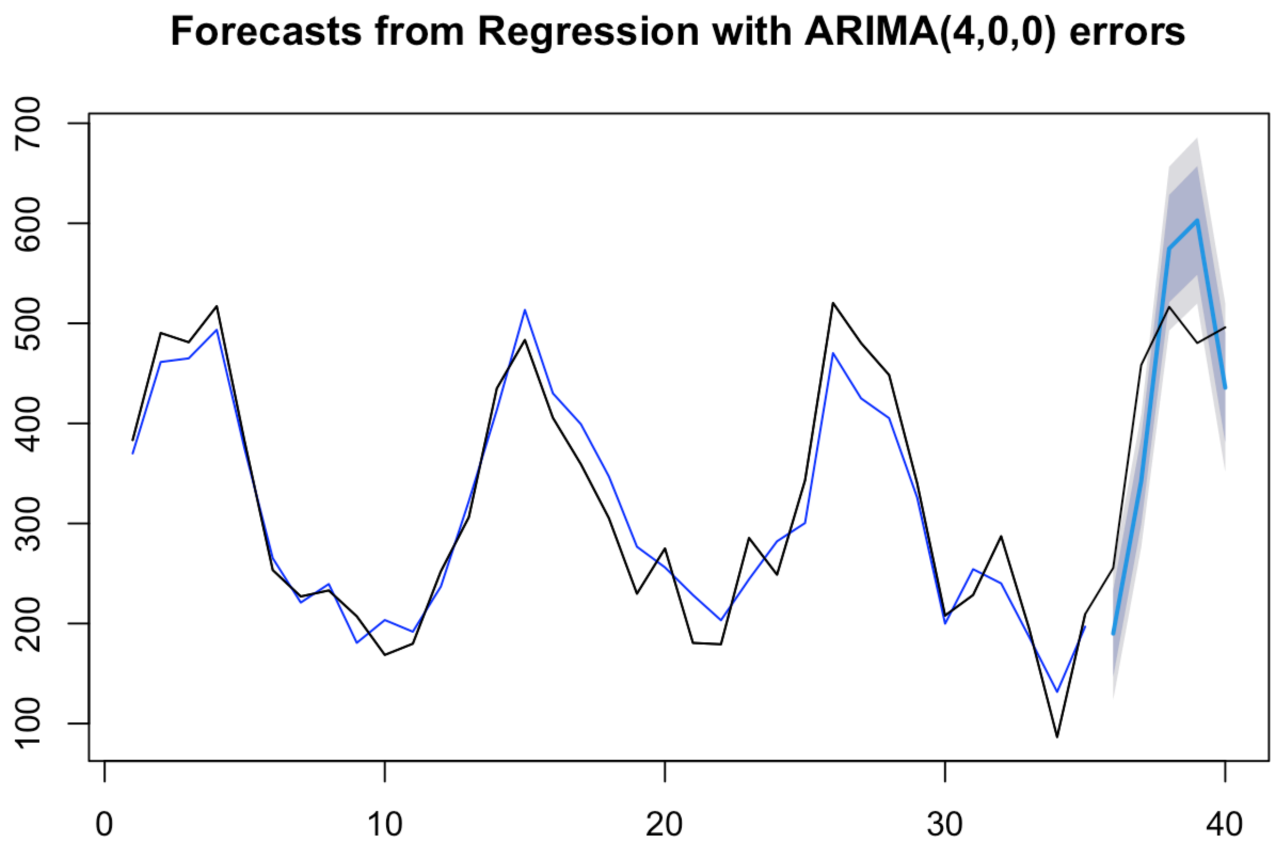
```
newpred <- as.matrix(cbind(predictor, predictor^2))

colnames(newpred) <- c("Pred", "Pred2")

ARIMA.with.Pred2 <- auto.arima(train, xreg = newpred, stepwise = FALSE)

ARIMA.with.Pred2
```

```
## Series: train
## Regression with ARIMA(4,0,0) errors
##
## Coefficients:
##          ar1      ar2      ar3      ar4    Pred   Pred2
##       0.0382  -0.7339  -0.1458  -0.7136  0.6933  0.0602
## s.e.  0.1215   0.1201   0.1202   0.1153  0.2998  0.0044
##
## sigma^2 estimated as 1133:  log likelihood=-171.58
## AIC=357.16   AICc=361.31   BIC=368.05
```

```
tsdisplay(residuals(ARIMA.with.Pred2), lag.max = 15, main = "Resid. Diagnostics AR(4)
Quadratic")
```



```
test.pred <- as.matrix(cbind(AvgTemp[36:40], AvgTemp[36:40]^2))

colnames(test.pred) <- c("Pred", "Pred2")

plot(forecast(ARIMA.with.Pred2, h = 5, xreg = test.pred))

points(1:length(train), fitted(ARIMA.with.Pred2), type = "l", col = "blue")

points(1:40, Bill, type = "l")
```



To compare models we can use the accuracy function. You simply store your forecast results in an object and then provide the true test set responses. Numerous metrics are then provided RootMSE is the closest to ASE that we have seen in SAS previously. Check out the R help documentation for references on the differenct prediction accuracy metrics.

```
# casts.avgtemp <- forecast(ARIMA.with.Pred,
#                           h = 5,
#                           xreg = data.frame(predictor = AvgTemp[36:40]))

casts.avgtemp <- forecast(ARIMA.with.Pred, h = 5, xreg = AvgTemp[36:40])

accuracy(casts.avgtemp, Bill[36:40])
```

```
##                      ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set  1.837313 30.67513 27.18185 -1.377615 10.48192 0.3929667 0.1406363
## Test set     18.592012 92.59637 89.16969  7.253808 21.57989 1.2891217        NA
```

```
cast.avgtemp.quad <- forecast(ARIMA.with.Pred2, h = 5, xreg = test.pred)

accuracy(cast.avgtemp.quad, Bill[36:40])
```

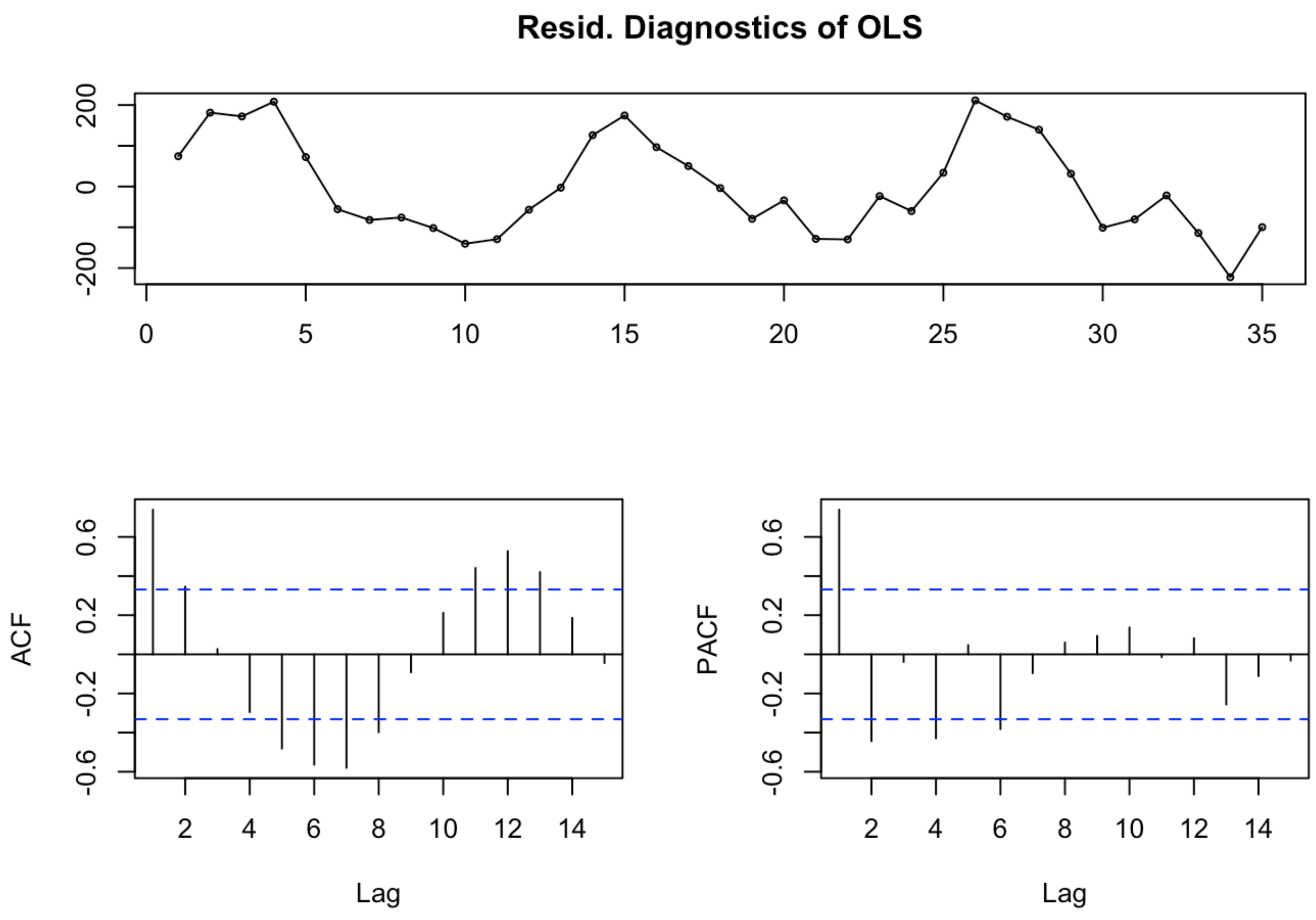
```
##                      ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set  1.717535 30.64328 26.88489 -1.975446 10.26099 0.3886736 0.1834575
## Test set     12.186182 89.18941 84.51464  5.260677 19.97953 1.2218239        NA
```

Taking this with a grain of salt since the test set only has 5 observations, the accuracy metrics on the test set are all lower for the quadratic term indicating that an improvement has been made although it is not drastic. Compared to how well the training fit is, it looks like we have some work to do.

Homework Exercises

14. Use the same train/test split previously discussed to fit an AR(4) model (do not include the predictor for AvgTemp) to train and forecast the last 5 observations. Provide a graphic of the forecast and produce the accuracy results on the 5 test set observations for comparison of the previous fits which included a predictor. Bonus question….why should we be careful, when comparing test accuracy metrics from models that are assumed stationary (AR4) and those that are not stationary (models with predictors)? Does it matter how many observations I include in the test set? Think about your result in 13.

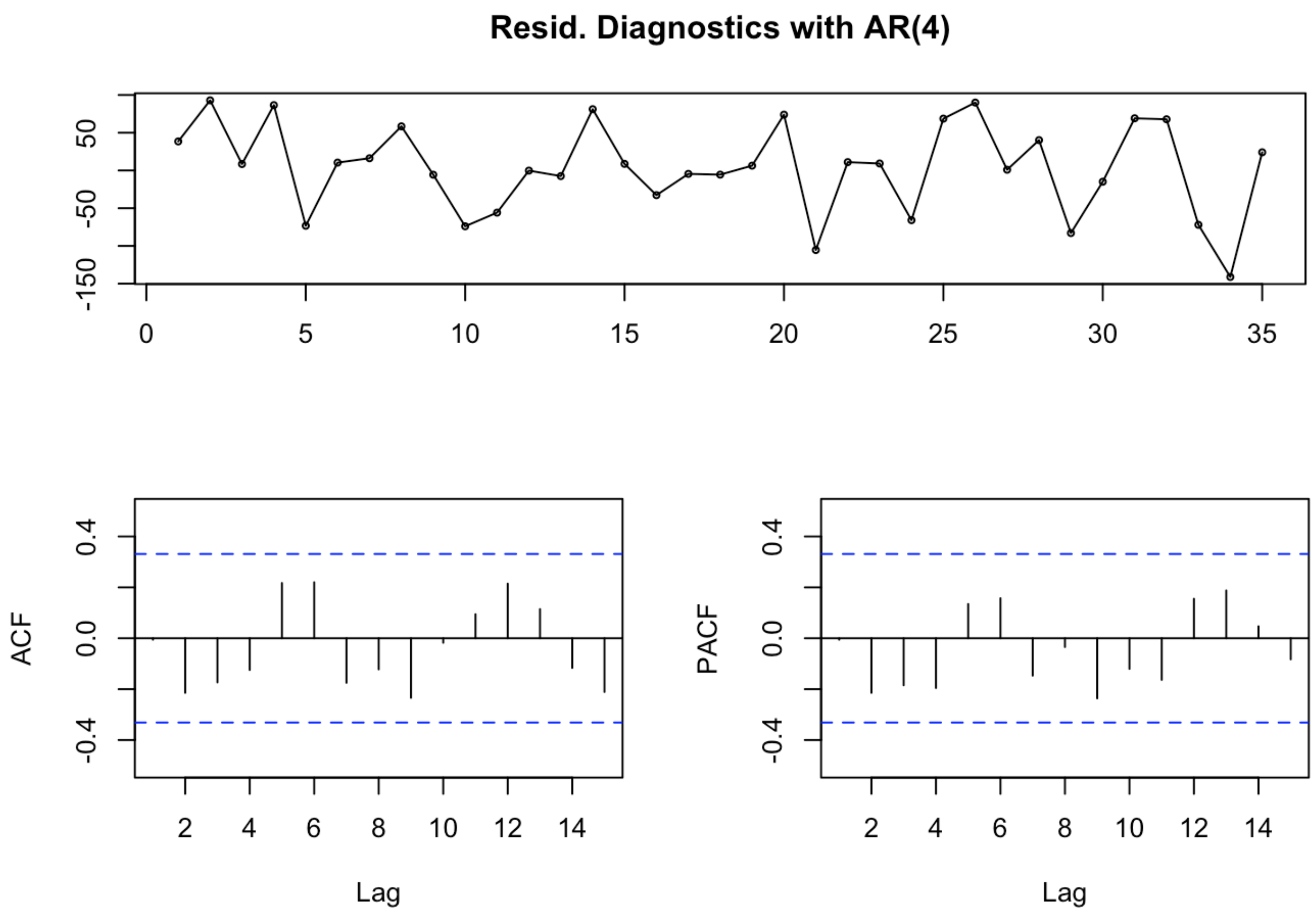
```
# Define holdout test
holdout.test <- window(ts(Bill), start = 36)

# Define training data
train <- Bill[1:35]

# Analyze residual diagnostics of OLS
simpleols <- arima(train, order = c(0, 0, 0))
tsdisplay(residuals(simpleols), lag.max = 15, main = "Resid. Diagnostics of OLS")
```



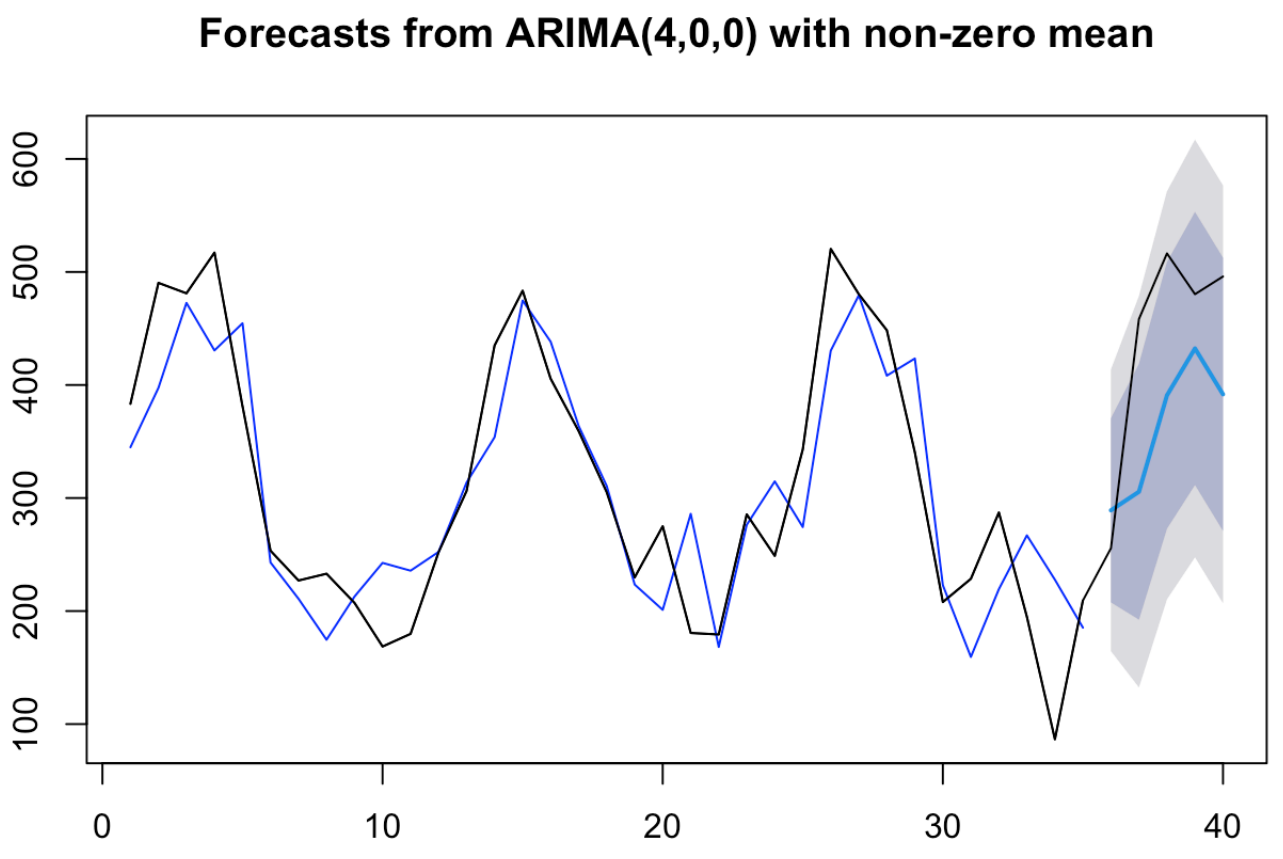
```
# Create ARIMA
ARIMA.wo.Pred <- auto.arima(train, stepwise = FALSE)

# Examine ARIMA
ARIMA.wo.Pred
```

```
## Series: train
## ARIMA(4,0,0) with non-zero mean
##
## Coefficients:
##          ar1      ar2     ar3      ar4      mean
##       0.9647  -0.5216  0.4276  -0.5403  304.8543
## s.e.  0.1453   0.2285  0.2214   0.1531   15.5346
##
## sigma^2 estimated as 4035:  log likelihood=-193.65
## AIC=399.3   AICc=402.3   BIC=408.63
```

```
# Plot Residual Diagnostics
tsdisplay(residuals(ARIMA.wo.Pred),
          lag.max = 15,
          main = "Resid. Diagnostics with AR(4)")
```



```
# Plot forecast
plot(forecast(
  ARIMA.wo.Pred,
  h = 5
))
points(1:length(train),
       fitted(ARIMA.wo.Pred),
       type = "l",
       col = "blue")

points(1:40, Bill, type = "l")
```



```
# Score forecast results
casts <- forecast(ARIMA.wo.Pred, h = 5)
accuracy(casts, Bill[36:40])
```

```
##                      ME      RMSE      MAE       MPE     MAPE      MASE
## Training set  3.450165  58.81251 45.83423 -4.912641 19.48712 0.6626231
## Test set     79.414280 103.36026 92.83666 15.106762 20.35870 1.3421350
##                      ACF1
## Training set -0.005700582
## Test set               NA
```

Differencing data to create a stationary model.

> No More Assignments Past this point.

The one thing that the HW doesn't cover so far is the idea of taking a first order time series to render a stationary time series. One way to do this is to use the ?diff. This makes it a little challenging to make forecasts back on the original scale. The way to do this using the tools already covered is realizes that an

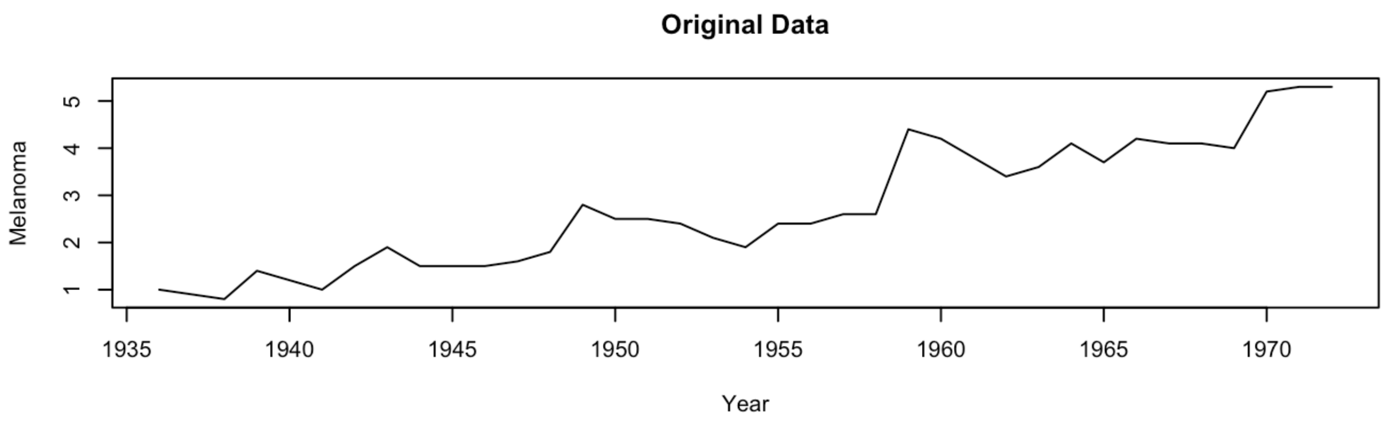
ARIMA(0,1,0), produces residuals that follow the same 1st order difference pattern. They are just centered at 0 instead of the original mean.

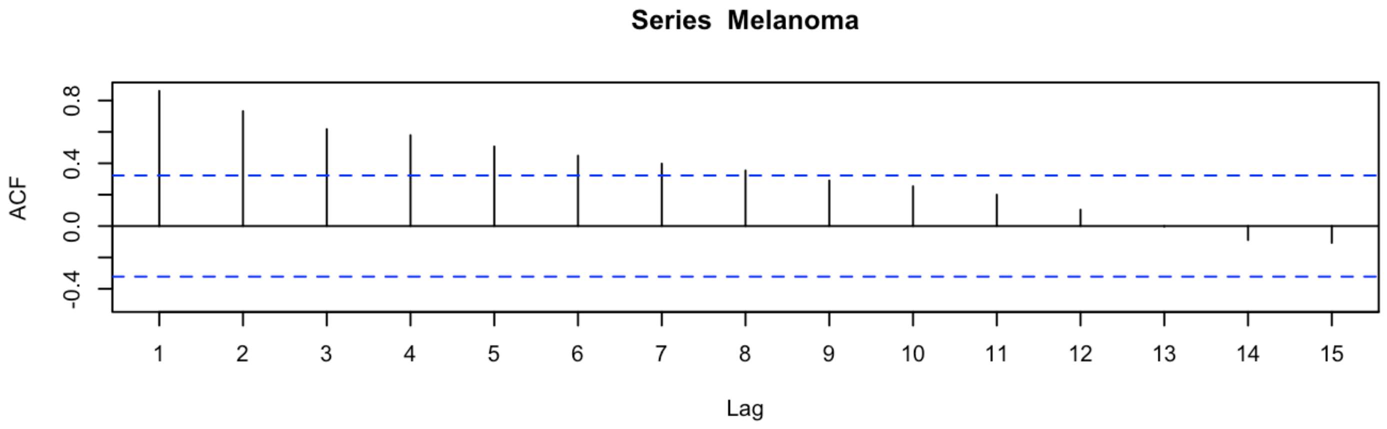
To illustrate, lets consider the melanoma time series from pre live work.

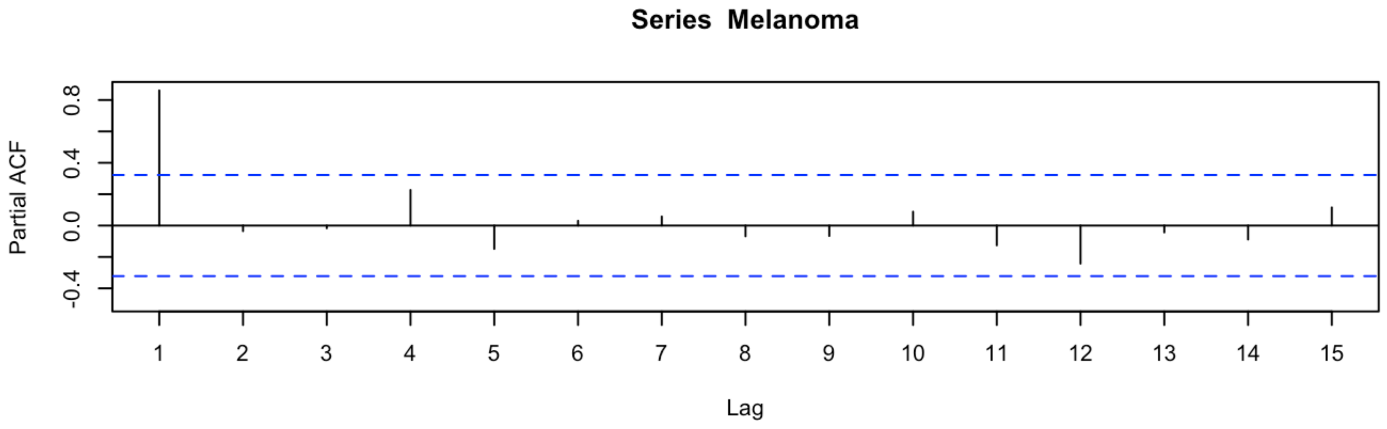
```
# Load melanoma data
melanoma <- read_csv(here::here("6372", "week 04", "Melanomatimeseries.csv"))
```

```
## Parsed with column specification:
## cols(
##   Year = col_double(),
##   Melanoma = col_double(),
##   Sunspot = col_double()
## )
```

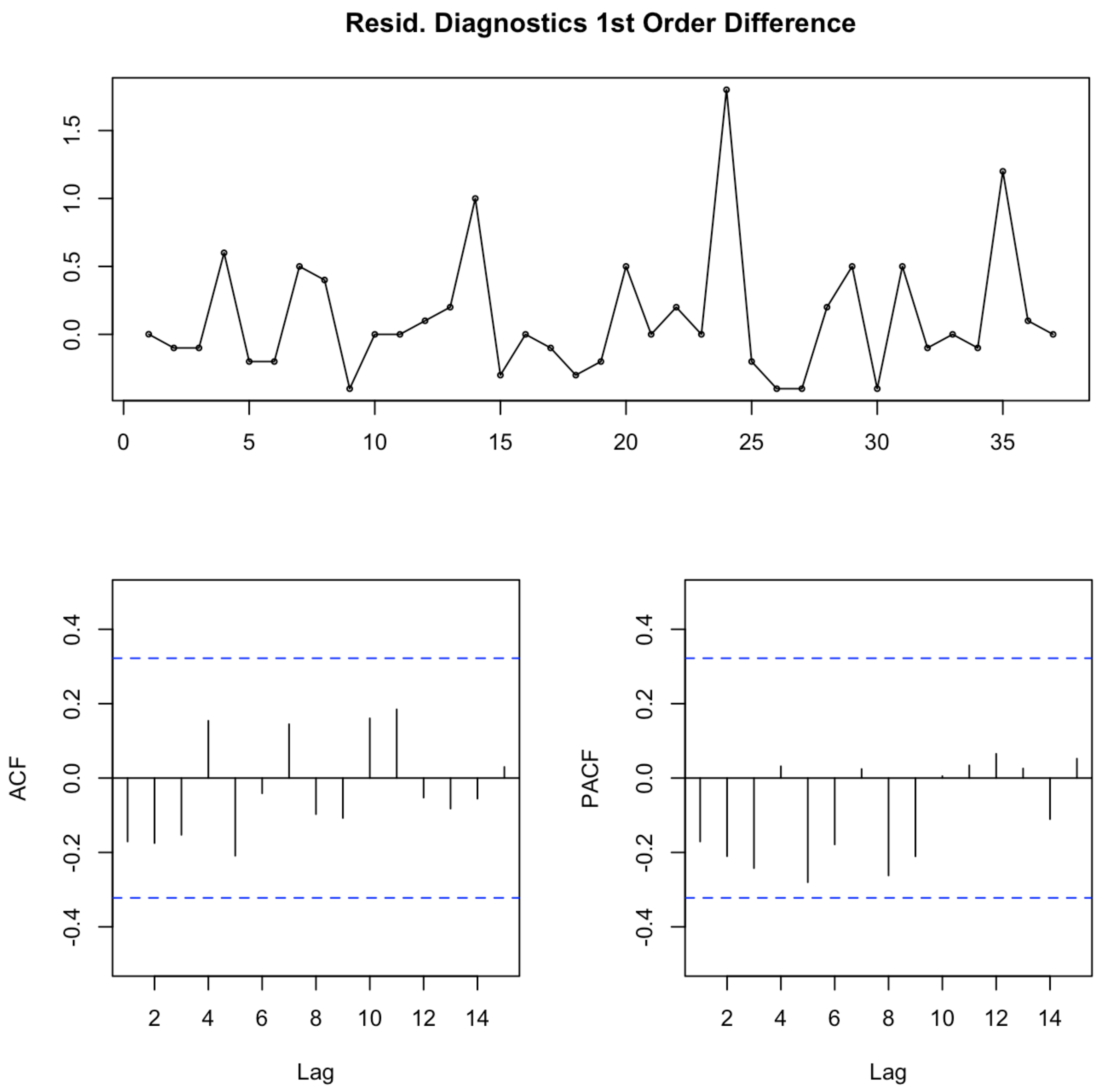
```
attach(melanoma)

par(mfrow = c(3, 1))

plot(Year, Melanoma, type = "l", main = "Original Data")

Acf(Melanoma)

Pacf(Melanoma)
```







```
par(mfrow = c(1, 1))

diff.data <- arima(Melanoma, order = c(0, 1, 0))

tsdisplay(residuals(diff.data),
          lag.max = 15,
          main = "Resid. Diagnostics 1st Order Difference")
```



We can see that by differencing the data, not only that we have a stationary time series, but there really is no serial correlation left to model! In many situations serial correlation will still exist after differencing the data, and the previous techniques can be used to model it out.