Name: Your name here
Due: 2024/11/20

Day 20 - Lab: ARMA models

Introduction

In this assignment, we will get some more practice with analyzing real data using time series models that incorporate ARMA(p, q) processes. In this lab, we will return to the electricity data that were introduced when decomposing time series at the start of the courses. As a reminder, this data set describes the monthly supply of electricity (in millions of kWh) in Australia over the period of January 1958 to December of 1990, according to the Australian Bureau of Statistics. The code below reads in the series.

```
cbe <- read_delim("cbe.dat")
elec <- dplyr::select(cbe, elec)
```

---

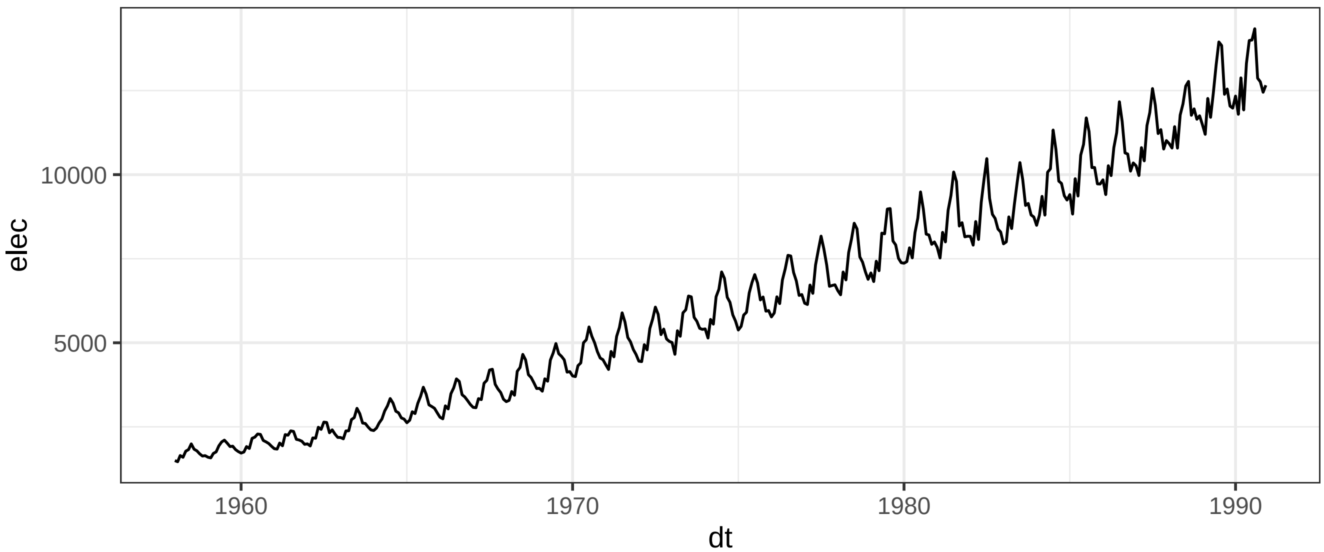
1. [2 pt] Create a data frame that includes the electricity measurement, the month, a scaled index for time (to have mean 0 and standard deviation 1), and the scaled time index squared.

```
elect_tbl <- elec %>%
  mutate(
    t = 1:n(),
    scaled_t = c(scale(t)),
    scaled_t2 = c(scaled_t^2),
    month = rep(1:12, length(1958:1990)) %>% factor(),
    year = rep(1958:1990, each = 12),
    dt = ym(paste0(year, "-", month))
  )
elect_tbl
```

```
# A tibble: 396 x 7
    elec     t scaled_t scaled_t2 month  year dt
   <dbl> <int>    <dbl>     <dbl> <fct> <int> <date>
 1  1497     1    -1.73      2.98 1      1958 1958-01-01
 2  1463     2    -1.72      2.95 2      1958 1958-02-01
 3  1648     3    -1.71      2.92 3      1958 1958-03-01
 4  1595     4    -1.70      2.89 4      1958 1958-04-01
 5  1777     5    -1.69      2.86 5      1958 1958-05-01
 6  1824     6    -1.68      2.83 6      1958 1958-06-01
 7  1994     7    -1.67      2.80 7      1958 1958-07-01
 8  1835     8    -1.66      2.77 8      1958 1958-08-01
 9  1787     9    -1.66      2.74 9      1958 1958-09-01
10  1699    10    -1.65      2.71 10     1958 1958-10-01
# i 386 more rows
```

2. [2 pt] Plot electricity over time and describe the series in terms of trend and seasonality.

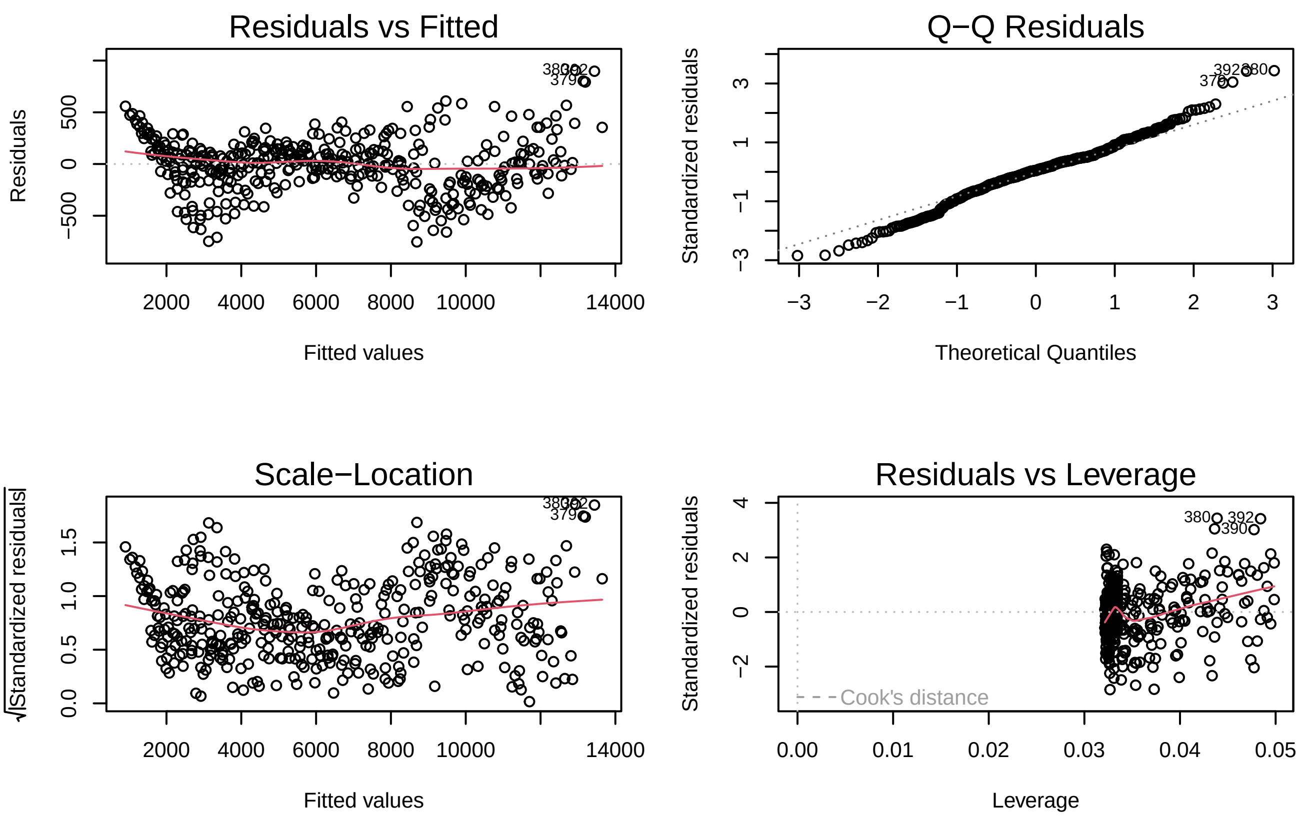
```
elect_tbl %>% ggplot() + geom_line(aes(x = dt, y = elec)) + theme_bw()
```

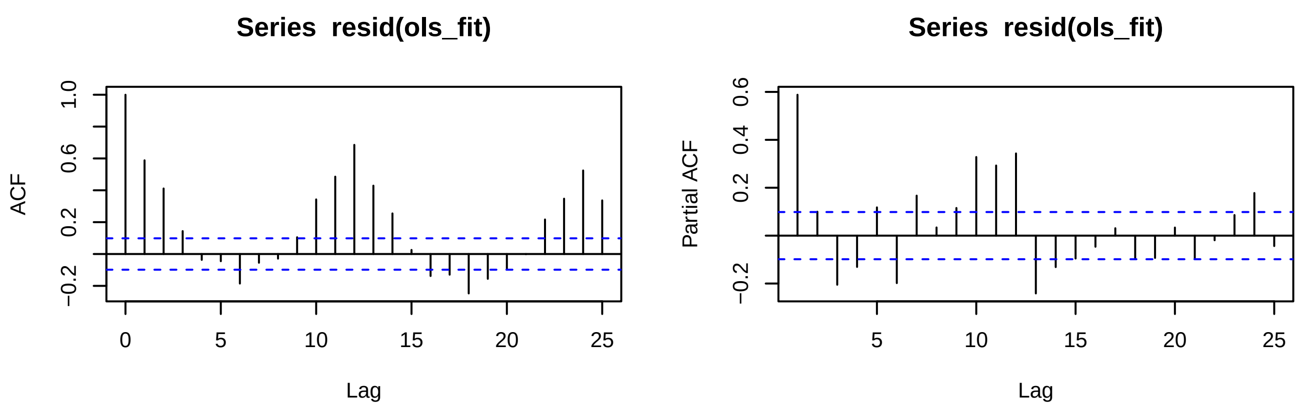


There seems to be an increasing quadratic relationship with time, and a clear seasonal effect in which electricity peaks during June through August.

3. [4 pt] Fit a regression model of the form `elect ~ scaled_t + scaled_t2 + month` and assess the assumptions for the fitted model.

```
ols_fit <- lm(elec ~ scaled_t + scaled_t2 + month, elect_tbl)
par(mfrow = c(3, 2))
plot(ols_fit)
acf(resid(ols_fit))
pacf(resid(ols_fit))
```



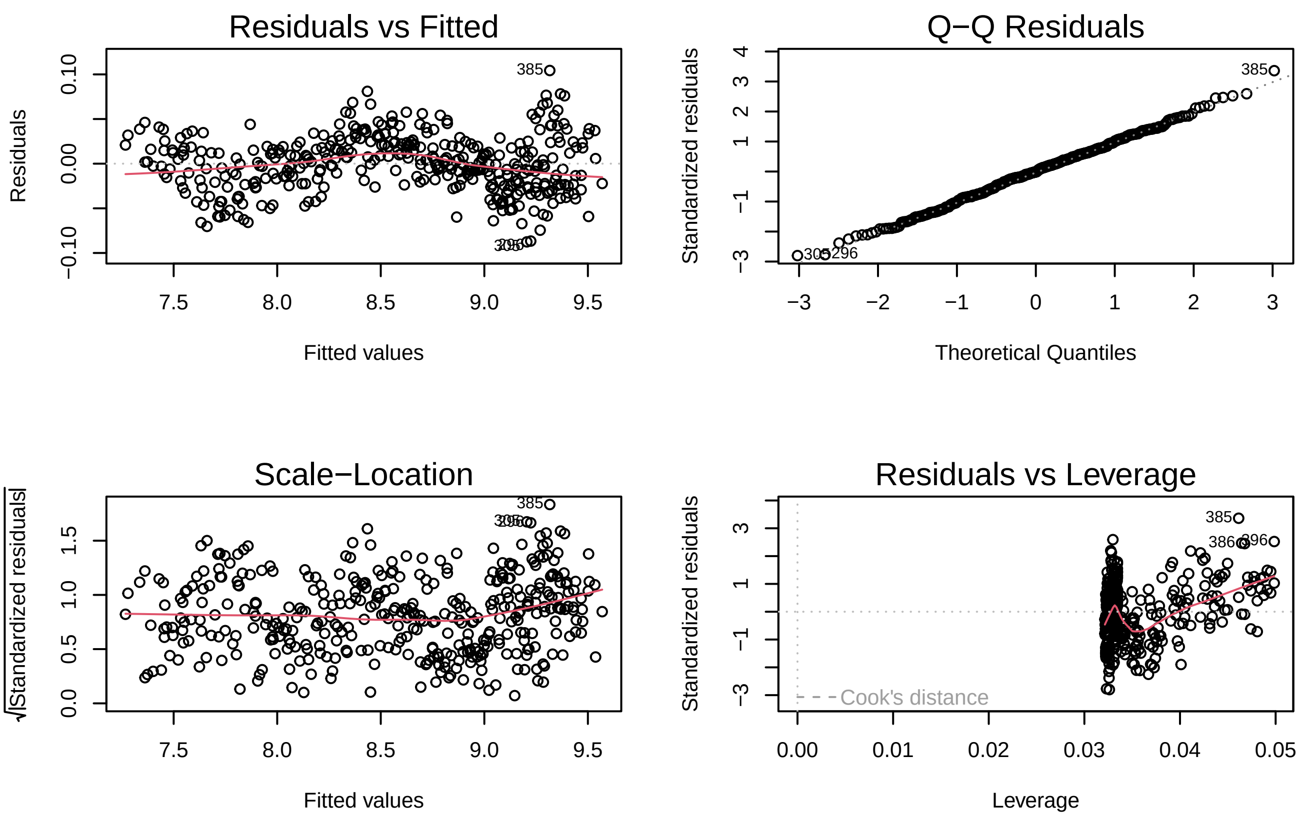


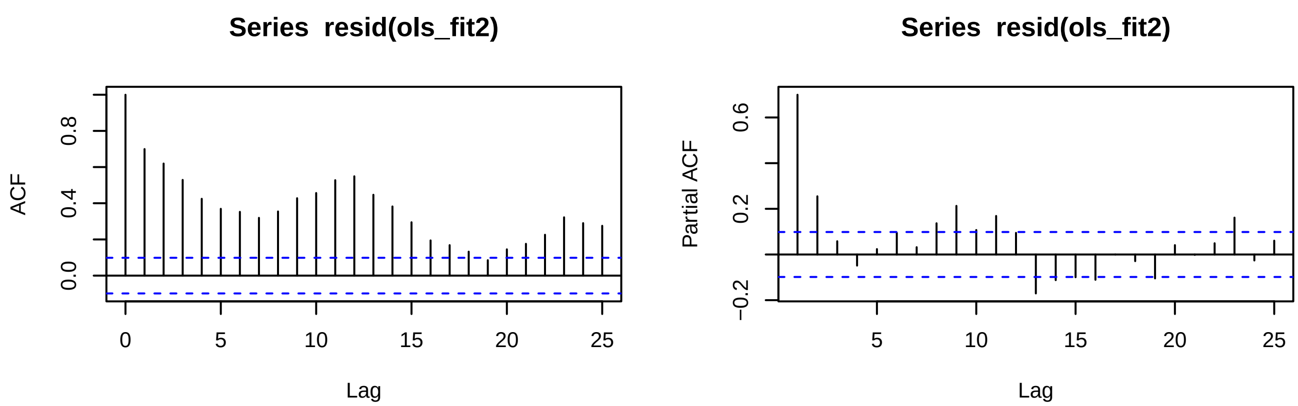
- Independence: The ACF and PACF plots suggest strong serial correlation in the residuals (at lags 1, 3, 6, 7, etc), so this assumption is violated.
- Constant variance: there appears to be some fanning in the residuals vs fitted plot, suggesting that this assumption is also violated
- Linearity: there is perhaps some leftover curvature in the residuals vs fitted plot, but it is hard to tell with the non-constant variance

- Normality: the points do not follow the hypothesized QQ line, meaning this assupmtion is also violated. However, with 360+ observations, the CLT will certainly provide approximately normally distributed sampling distributions of the regression coefficients.

4. [2 pt] Fit the same model again, this time using `log(elec)` as the response. Which assumptions are still violated?

```
ols_fit2 <- lm(log(elec) ~ scaled_t + scaled_t2 + month, elect_tbl)
par(mfrow = c(3, 2))
plot(ols_fit2)
acf(resid(ols_fit2))
pacf(resid(ols_fit2))
```

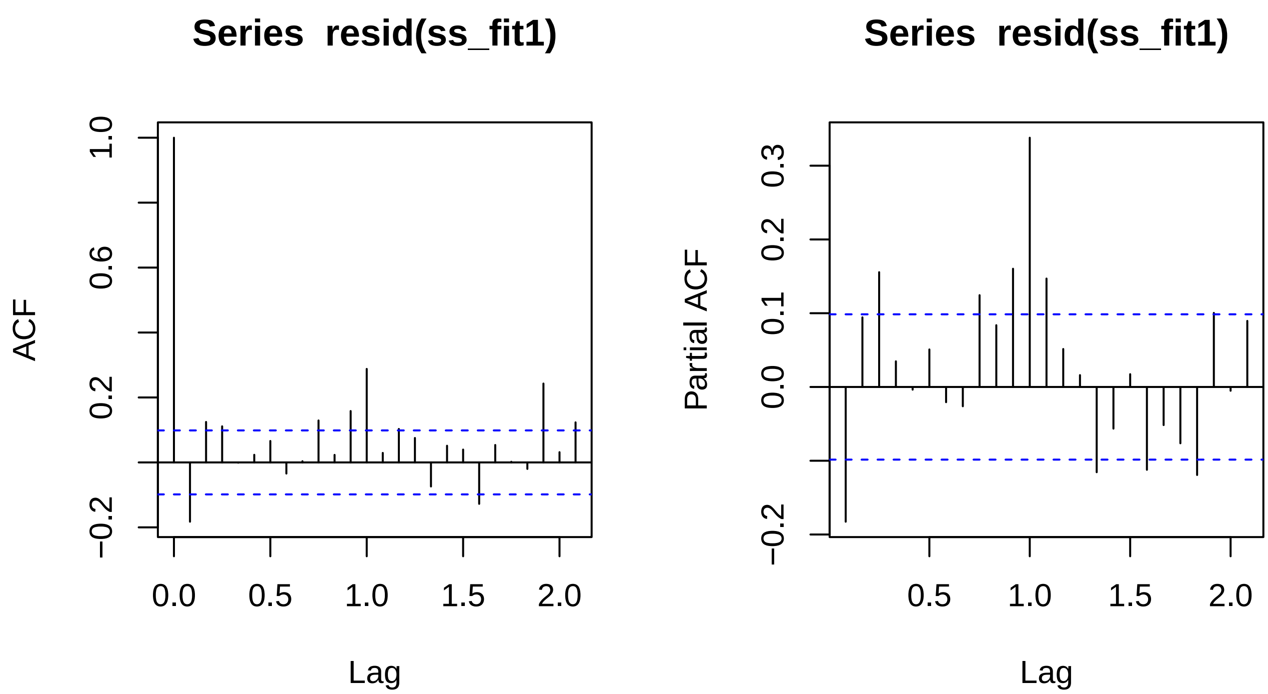




The log transformation helped with everything except linearity and independence. There is definitely some leftover curvature in the residuals vs fitted plot, and we still have serial correlation in the residuals.

5. [2 pt] Ignore any violations of the assumptions for now (except for independence) and use the log-transformed structure for all remaining questions. Use `arima` to fit the regression model with an AR(1) correlation structure. Assess the residual serial correlation.

```r
elect_ts <- ts(
  log(elect_tbl$elec),
  start = c(1958, 1),
  freq = 12
)
ss_fit1 <- arima(
  elect_ts,
  order = c(1, 0, 0),
  xreg = model.matrix(~ scaled_t + scaled_t2 + month, elect_tbl),
  include.mean = F
)
par(mfrow = c(1, 2))
acf(resid(ss_fit1))
pacf(resid(ss_fit1))
```

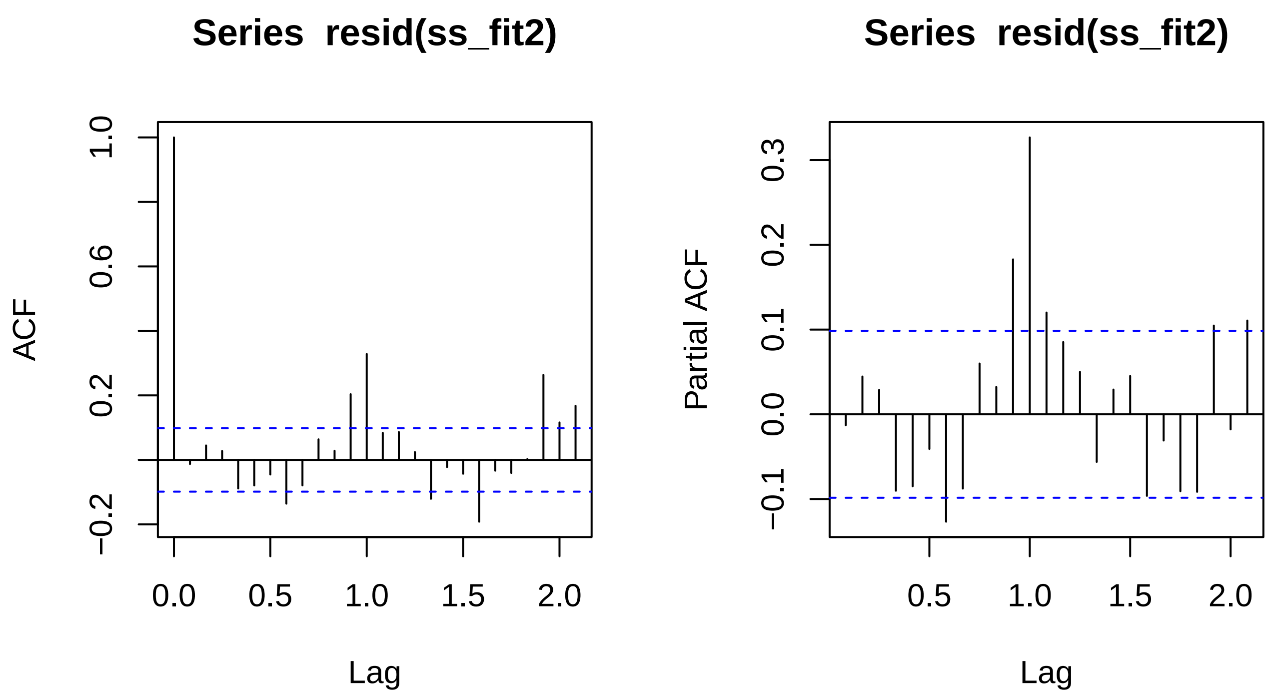


Still quite bad! Lots of serial correlation at multiple lags.

6. [2 pt] Use `arima` to fit the regression model with an ARMA(1, 1) structure. Assess the residual serial correlation.

```r
ss_fit2 <- arima(
  elect_ts,
```

```
  order = c(1, 0, 1),
  xreg = model.matrix(~ scaled_t + scaled_t2 + month, elect_tbl),
  include.mean = F
)
par(mfrow = c(1, 2))
acf(resid(ss_fit2))
pacf(resid(ss_fit2))
```



> Better, but still residual correlation at multiple lags. Notably, adding the MA(1) component removed the residual correlation that exists at lag 1 (which should make sense!)

7. [4 pt] Use the `auto.arima` function in the `forecast` package to pick the best non-seasonal ARMA model (set `max.d, max.D, max.P, max.Q` all equal to 0) for this regression model and print off the fit. Assess the residual serial correlation. You will need to set `allowmean = F` and `allowdrift = F` in the function call, since both the mean and the drift are included in our regression model (the y-intercept and the slope with time).

```
library(forecast, quietly = T)
```

```
Warning: package 'forecast' was built under R version 4.3.3
```

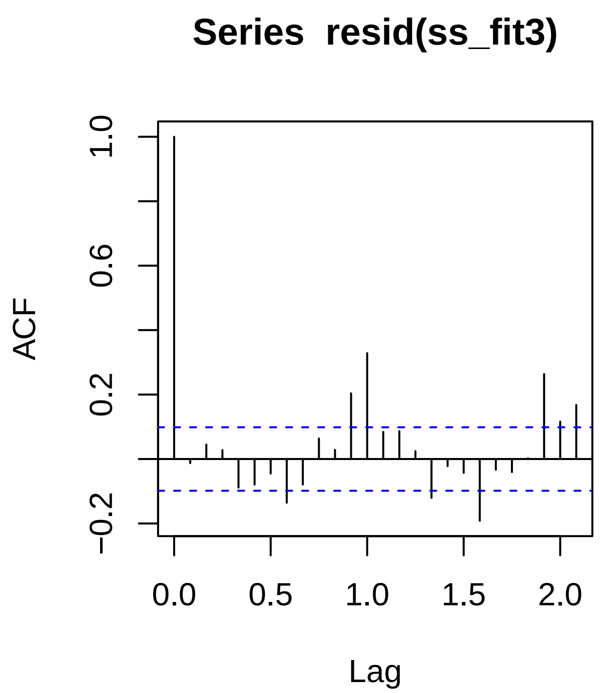
```
Registered S3 method overwritten by 'quantmod':
```

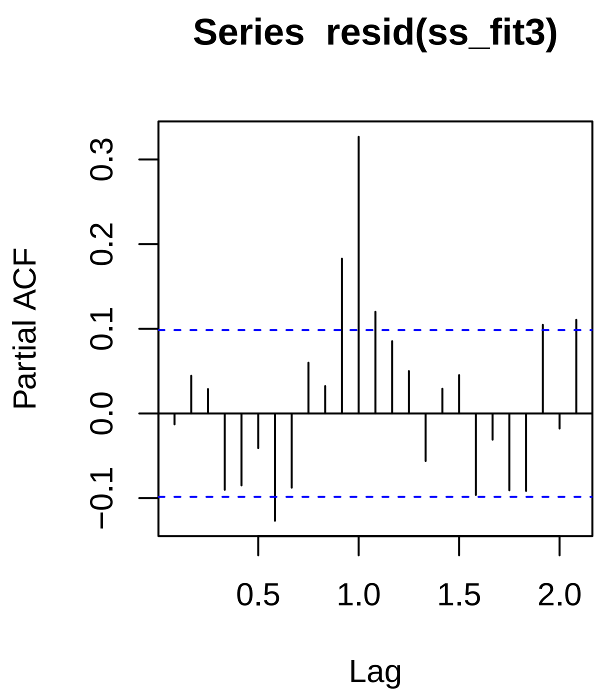
```
  method             from
  as.zoo.data.frame zoo

Attaching package: 'forecast'

The following object is masked from 'package:nlme':

    getResponse
```

```
ss_fit3 <- auto.arima(
  elect_ts,
  max.d = 0,
  max.D = 0,
  max.P = 0,
  max.Q = 0,
  xreg = model.matrix(~ scaled_t + scaled_t2 + month, elect_tbl),
  allowmean = F,
  allowdrift = F
)
ss_fit3
```

```
Series: elect_ts
Regression with ARIMA(1,0,1) errors

Coefficients:
         ar1      ma1  (Intercept)  scaled_t  scaled_t2   month2  month3
      0.8762  -0.3487       8.5786    0.5994    -0.0865  -0.0201  0.0656
s.e.  0.0325   0.0619       0.0090    0.0054     0.0059   0.0041  0.0047
      month4  month5  month6  month7  month8  month9  month10  month11  month12
      0.0324  0.1457  0.1771  0.2367  0.1986  0.1065   0.0896   0.0419   0.0227
s.e.  0.0050  0.0053  0.0054  0.0055  0.0054  0.0053   0.0050   0.0047   0.0042

sigma^2 = 0.0004746:  log likelihood = 961.15
AIC=-1888.29   AICc=-1886.67   BIC=-1820.61
```

```
par(mfrow = c(1, 2))
acf(resid(ss_fit3))
pacf(resid(ss_fit3))
```





It chooses the same ARMA(1,1) model! So just the same as before. Looks like we cannot do much better within the ARMA(p, q) framework...

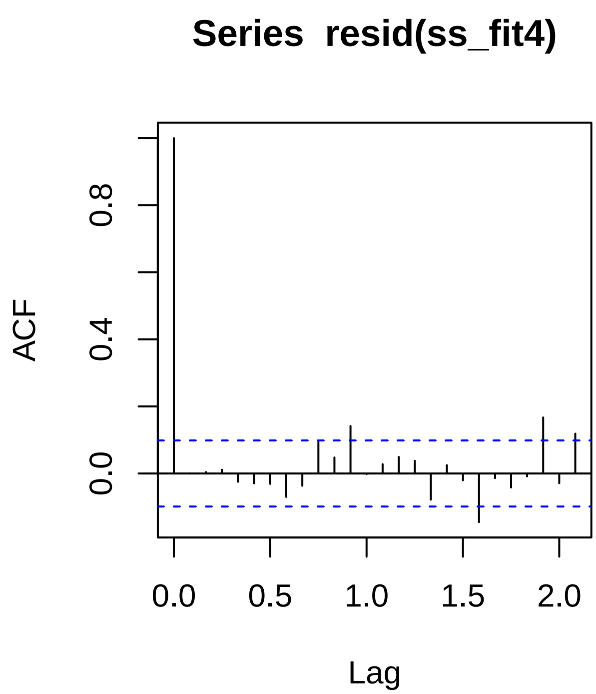
8. [4 pt] Allow `auto.arima` to choose any form for the errors in the regression model by no longer setting `max.d, max.D, max.P, max.Q` equal to 0 (might take a second or two). You will again need to turn off the drift and mean parameters. Print off the fit and assess the residual correlation.

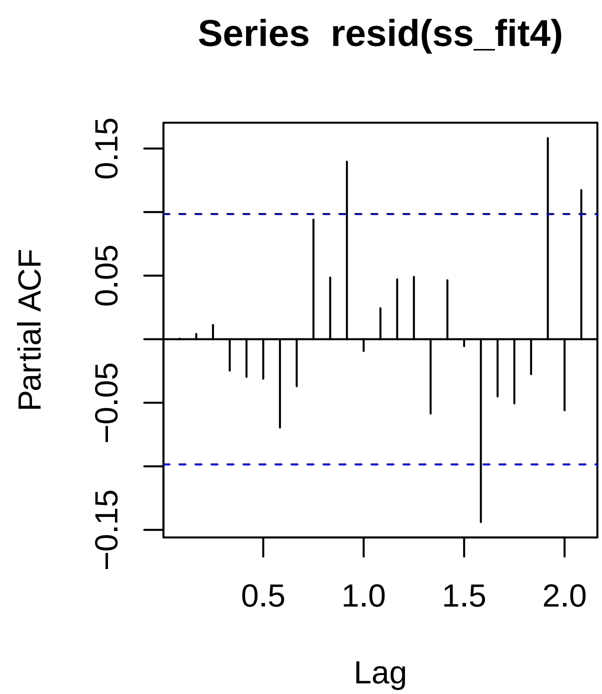
```
ss_fit4 <- auto.arima(
  elect_ts,
  xreg = model.matrix(~ scaled_t + scaled_t2 + month, elect_tbl),
  allowmean = F,
  allowdrift = F
)
ss_fit4
```

```
Series: elect_ts
Regression with ARIMA(2,0,1)(2,0,0)[12] errors

Coefficients:
         ar1     ar2      ma1    sar1    sar2  (Intercept)  scaled_t  scaled_t2
      0.7725  0.0907  -0.4043  0.3810  0.0619       8.5772    0.5998    -0.0830
s.e.  0.1382  0.0999   0.1423  0.0516  0.0579       0.0124    0.0069     0.0073
       month2  month3  month4  month5  month6  month7  month8  month9  month10
```

```
      -0.0209  0.0654  0.0311  0.1437  0.1754  0.2340  0.1961  0.1050   0.0874
s.e.   0.0071  0.0074  0.0079  0.0082  0.0083  0.0084  0.0083  0.0082   0.0079
      month11  month12
       0.0402   0.0222
s.e.   0.0075   0.0072

sigma^2 = 0.0004114:  log likelihood = 989.89
AIC=-1939.78   AICc=-1937.54   BIC=-1860.15
```

```
par(mfrow = c(1, 2))
acf(resid(ss_fit4))
pacf(resid(ss_fit4))
```





We are getting close! Still some leftover correlation at later lags (3 in particular: lags 11, 17, and 19)

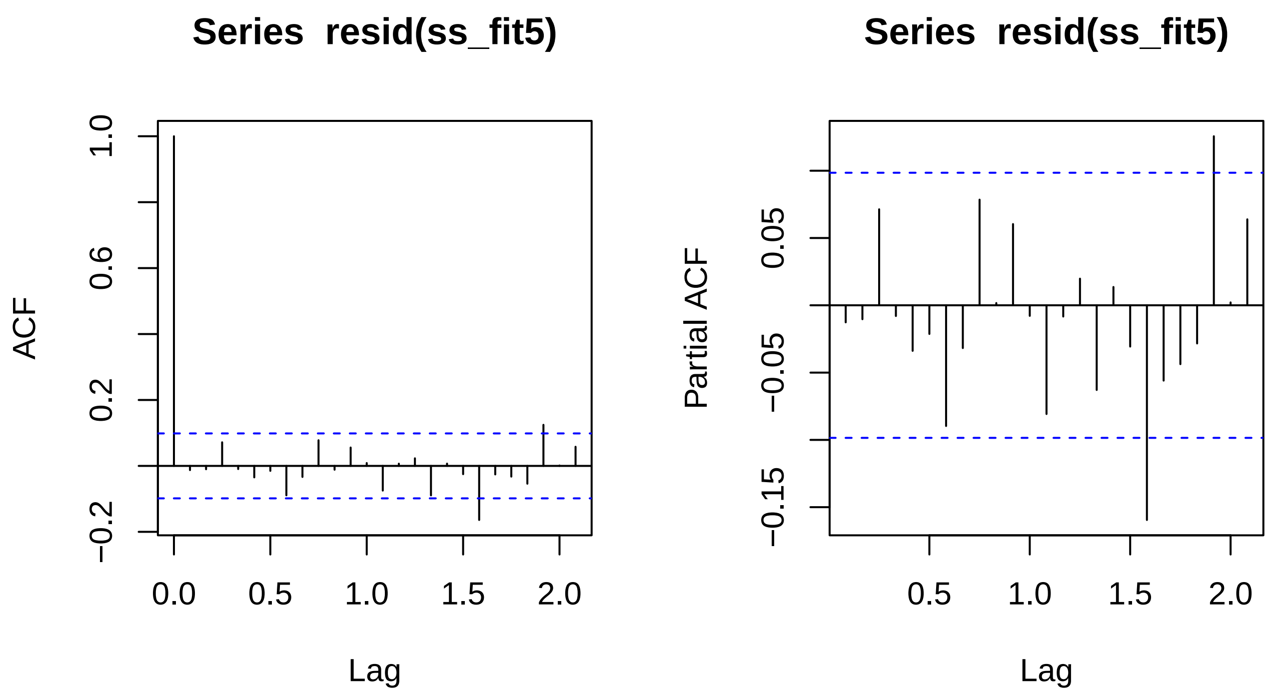
9. [4 pt] Finally, omit the regression model all together and allow `auto.arima` to select the best model. Print off the model and assess the residual correlation.

```
ss_fit5 <- auto.arima(
  elect_ts
)
```

```r
ss_fit5
```

```
Series: elect_ts
ARIMA(2,1,1)(2,1,2)[12]

Coefficients:
         ar1     ar2      ma1      sar1    sar2     sma1      sma2
      0.1484  0.1216  -0.7888  -0.4863  0.0219  -0.1150  -0.4971
s.e.  0.1188  0.0887   0.0953   0.2780  0.0998   0.2728   0.2234

sigma^2 = 0.0004251:  log likelihood = 942.85
AIC=-1869.71   AICc=-1869.32   BIC=-1838.12
```

```r
par(mfrow = c(1, 2))
acf(resid(ss_fit5))
pacf(resid(ss_fit5))
```



Pretty good now! Not perfect, but pretty good. The remaining correlation that exists at later lags is likely a function of information we do not have (perhaps the average winter temperature, for example)

10. [2 pt] We will learn about the model that `auto.arima` selected in question 8 and 9 next week. For

now, comment on the advantages and disadvantages of the purely stochastic approach implemented by `auto.arima` with no `xreg` relative to including predictor variables in a regression model.

> The stochastic approach is nice because it usually does a decent job accounting for serial correlation. However, you lose the ability to make inferences about potential variables of interest. For example, the stochastic approach prevents us from including other factors that might explain the electricity use, such as the average winter temperature, or something like that.