Lab Session 3: Time series modeling

Don Fejfar

3/16/2021

Set working directory

We will first set a working directory. Your working directory should be where your plan to save your R code and also where the example datasets have been stored. In the "Files" tab, navigate to this folder. Once you are there, select the "More" dropdown and select "Set As Working Directory".

Install and load R packages

You should have already installed the tidyverse package. Now, you will need to load the package into R. This will allow you to use the functionality of the package.

```
library(tidyverse)
library(lubridate)
library(car)
```

STEP 1. Choose relevant indicator and visualize data

Load the monthly number of acute respiratory infections ".rds" file and save it as a data frame called "facility".

```
facility <- readRDS("session3_data/example_facility_ari.rds") %>%
  # we are adding this code in here to format as date variable
  # this will fix the issue with the plots
  mutate(date=as.Date(date))
```

View the first six observations in the facility dataset.

```
head(facility)
```

```
## # A tibble: 6 x 2
##   date       ari_count
##   <date>         <dbl>
## 1 2016-01-01       275
## 2 2016-02-01       258
## 3 2016-03-01       249
## 4 2016-04-01       172
## 5 2016-05-01       230
## 6 2016-06-01       342
```

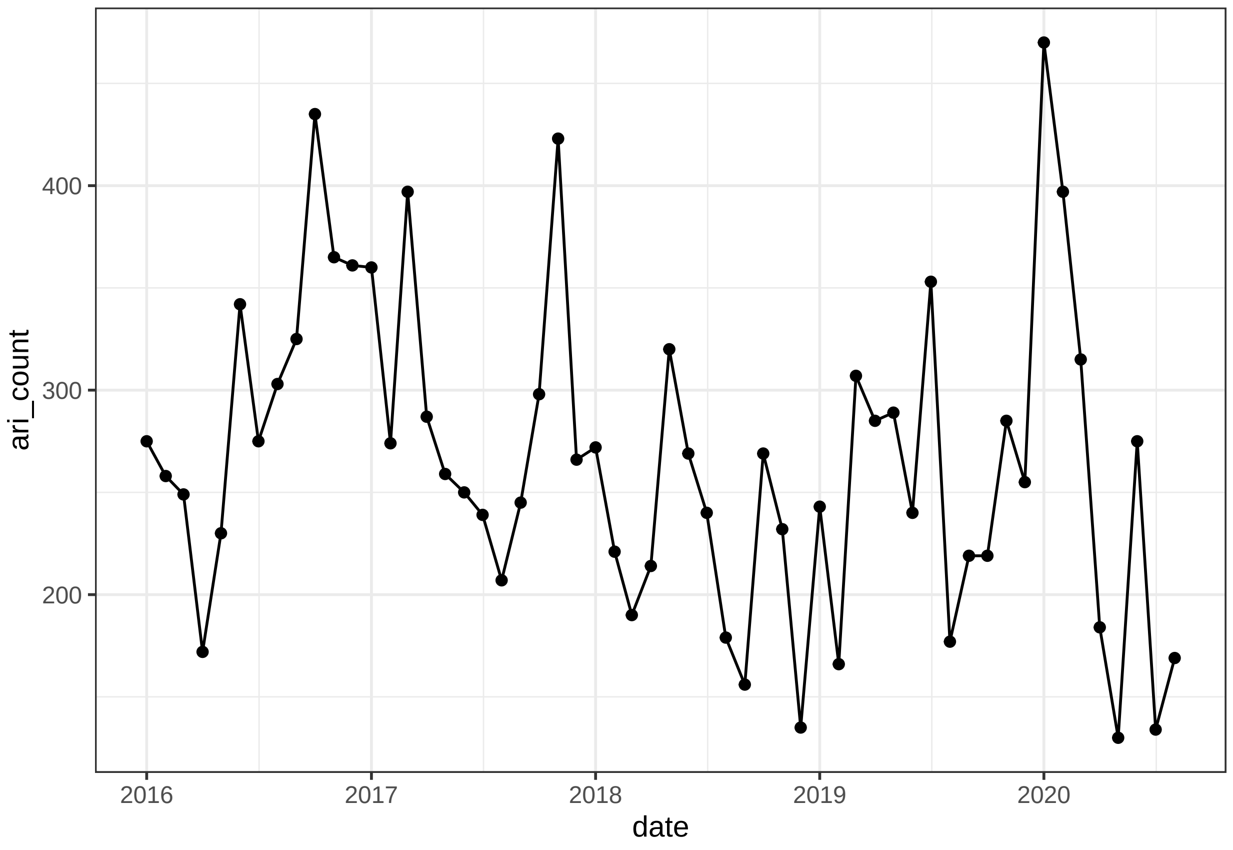
What is the date range in the dataset?

```
facility %>%
  summarize(min(date),
            max(date))
```

```
## # A tibble: 1 x 2
##   `min(date)` `max(date)`
##   <date>      <date>
## 1 2016-01-01  2020-08-01
```

Create a scatterplot of the date and actue respiratory infection count.

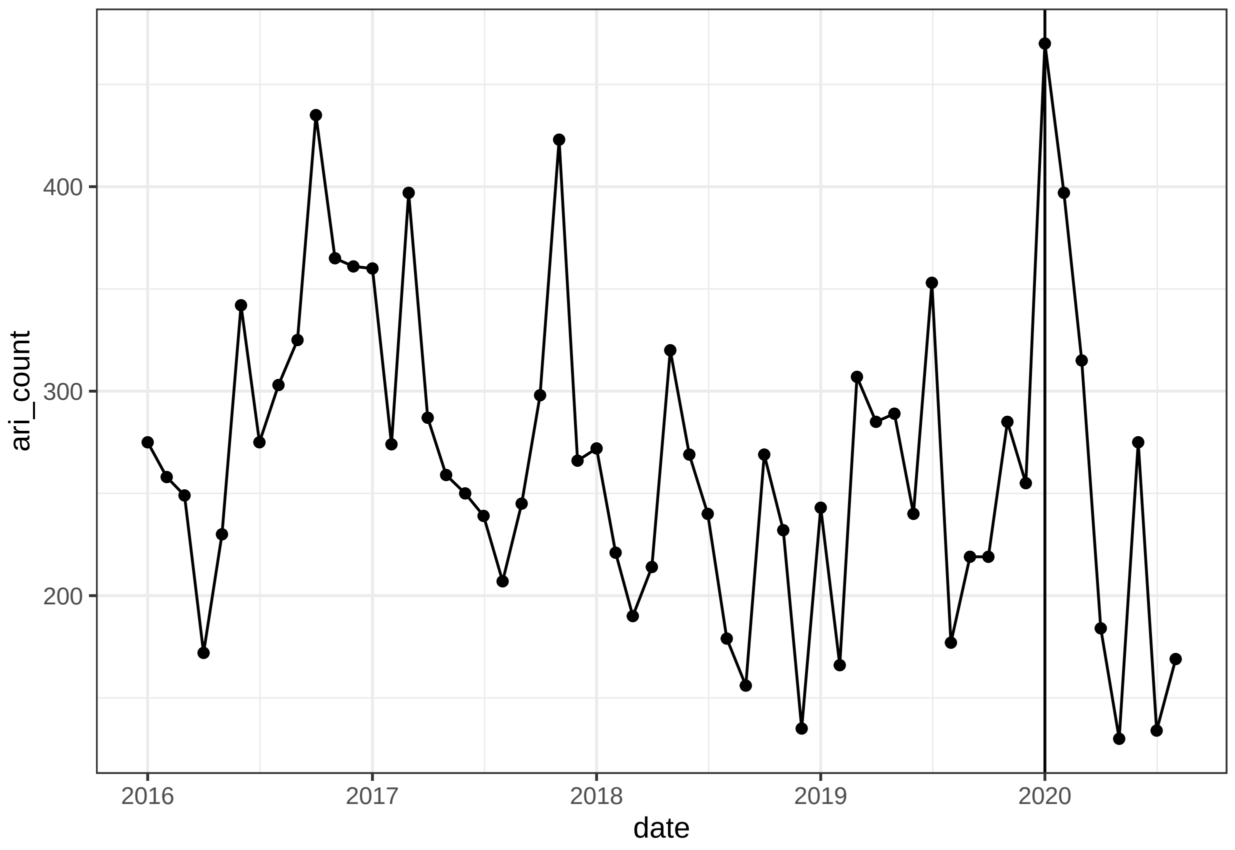
```r
ggplot(facility,aes(x=date,y=ari_count)) +
  geom_point() +
  geom_line() +
  theme_bw()
```



STEP 2. Choose the baseline and evaluation period

The evaluation period will start on January 1, 2020. Include a vertical line on the scatter plot.

```r
ggplot(facility,aes(x=date,y=ari_count)) +
  geom_point() +
  geom_line() +
  theme_bw() +
  geom_vline(xintercept=as.Date("2020-01-01"))
```



STEP 3. Fit time series model to baseline period

Modify the facility dataset to include: time (month number), year, one cosine, and one sine term.

```
facility %>%
  arrange(date) %>%
  mutate(time=1:n(),
         year=year(date)) %>%
  mutate(cos1=cos(2*pi*time/12),
         sin1=sin(2*pi*time/12)) -> facility_new
```

Fit a model with only a linear time term on the baseline period data. Call this *fit1*.

```
facility_new_baseline <- facility_new %>% filter(date < as.Date("2020-01-01"))

fit1 <- lm(ari_count ~ time,data=facility_new_baseline)

summary(fit1)
```

```
##
## Call:
## lm(formula = ari_count ~ time, data = facility_new_baseline)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -126.186  -43.408   -7.285   42.444  153.448
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 304.2145    18.9005  16.096   <2e-16 ***
## time         -1.5071     0.6715  -2.244   0.0297 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 64.45 on 46 degrees of freedom
## Multiple R-squared:  0.09868,    Adjusted R-squared:  0.07909
## F-statistic: 5.037 on 1 and 46 DF,  p-value: 0.02967
```

STEP 4. Using the model from Step 3, calculate deviations from expected in the evaluation period.

Calculate the difference between observed and expected in the evaluation period AND the 95% prediction intervals. Do you notice a pattern?

```
predicted_values <- predict(fit1, facility_new, interval="predict")

predicted_values %>%
  data.frame() %>%
  mutate(date=facility_new$date,
         observed=facility_new$ari_count) -> predicted_values_new


predicted_values_new %>%
  mutate(deviations=observed-fit) %>%
  filter(date >= as.Date("2020-01-01"))
```

```
##         fit      lwr      upr       date observed deviations
## 49 230.3688 95.16886 365.5687 2020-01-01      470  239.63121
## 50 228.8617 93.32437 364.3991 2020-02-01      397  168.13826
## 51 227.3547 91.46727 363.2421 2020-03-01      315   87.64532
## 52 225.8476 89.59765 362.0976 2020-04-01      184  -41.84763
## 53 224.3406 87.71563 360.9655 2020-05-01      130  -94.34057
## 54 222.8335 85.82129 359.8457 2020-06-01      275   52.16649
## 55 221.3265 83.91475 358.7382 2020-07-01      134  -87.32646
## 56 219.8194 81.99611 357.6427 2020-08-01      169  -50.81940
```

In order to compare the deviations across each month, standardize by the predicted values.

```
predicted_values_new %>%
  mutate(deviations=(observed-fit)/fit) %>%
  filter(date >= as.Date("2020-01-01"))
```

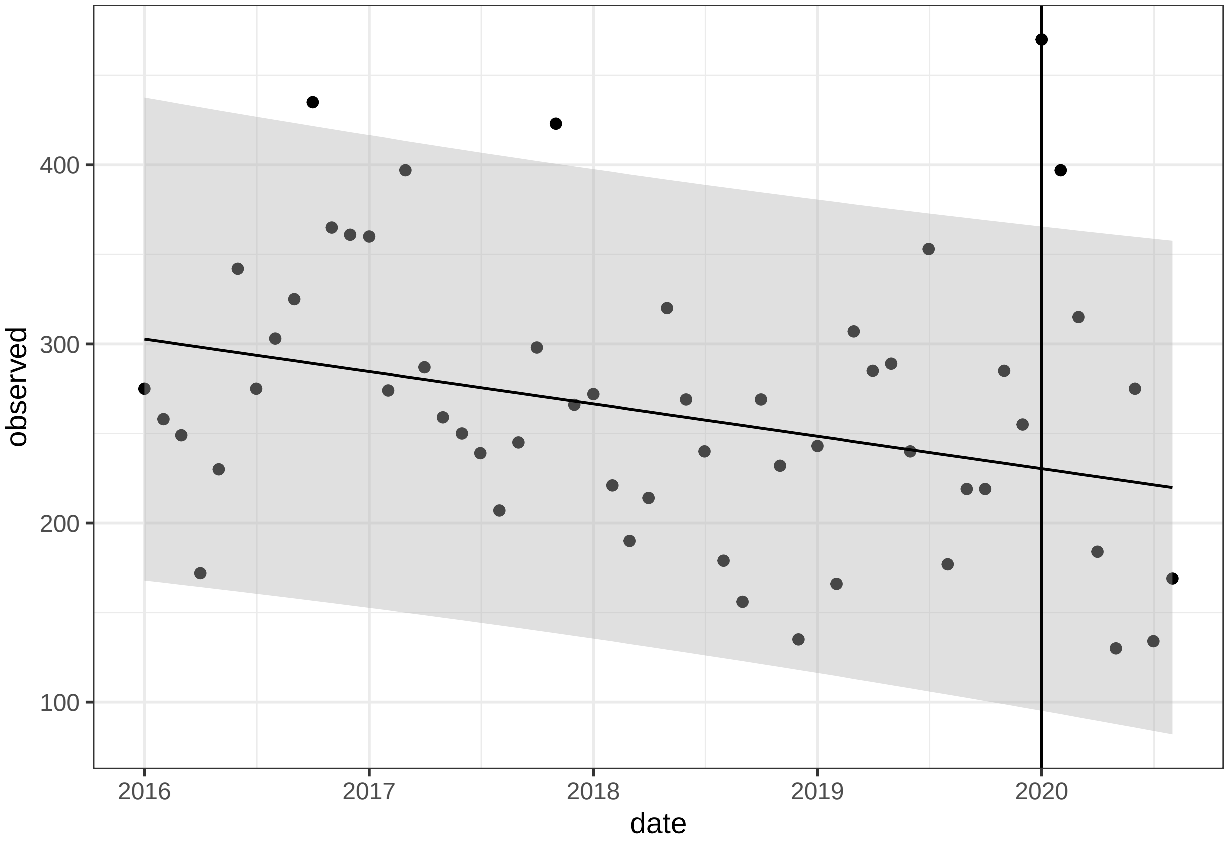
```
##         fit      lwr      upr       date observed deviations
## 49 230.3688 95.16886 365.5687 2020-01-01      470  1.0402069
## 50 228.8617 93.32437 364.3991 2020-02-01      397  0.7346718
## 51 227.3547 91.46727 363.2421 2020-03-01      315  0.3855004
## 52 225.8476 89.59765 362.0976 2020-04-01      184 -0.1852914
## 53 224.3406 87.71563 360.9655 2020-05-01      130 -0.4205239
## 54 222.8335 85.82129 359.8457 2020-06-01      275  0.2341052
## 55 221.3265 83.91475 358.7382 2020-07-01      134 -0.3945595
## 56 219.8194 81.99611 357.6427 2020-08-01      169 -0.2311871
```

Step 5. Produce interpretable visualizations.

Plot the observed values (points), predicted line, and 95% prediction intervals. **Hint:** You will need to pull the date and observed counts into the prediction interval data frame.

```
predicted_values %>%
  data.frame() %>%
  mutate(date=facility_new$date,
         observed=facility_new$ari_count) -> predicted_values_new

ggplot(predicted_values_new,aes(x=date,y=observed)) +
  geom_point() +
  geom_ribbon(aes(ymin=lwr,ymax=upr),fill = "grey70",alpha=.4) +
  geom_line(aes(x=date,y=fit)) +
  geom_vline(xintercept=as.Date("2020-01-01")) +
  theme_bw() +
  theme(legend.position="none")
```



ACTIVITY

1. Repeat Steps 3-5 above with a model including the following terms: year, cosine, and sine term (created above). Call this *fit2*.

```
# Fit the model
facility_new_baseline <- facility_new %>% filter(date < as.Date("2020-01-01"))

fit2 <- lm(ari_count ~ year + cos1 + sin1,data=facility_new_baseline)

summary(fit2)
```

```
##
## Call:
## lm(formula = ari_count ~ year + cos1 + sin1, data = facility_new_baseline)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -134.129  -37.585   -6.969   40.191  133.587
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 41693.292  16798.115   2.482   0.0170 *
## year          -20.533      8.326  -2.466   0.0176 *
## cos1           12.104     13.165   0.919   0.3629
## sin1           -2.744     13.165  -0.208   0.8358
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 64.49 on 44 degrees of freedom
## Multiple R-squared:  0.1368, Adjusted R-squared:  0.0779
## F-statistic: 2.324 on 3 and 44 DF,  p-value: 0.088
```

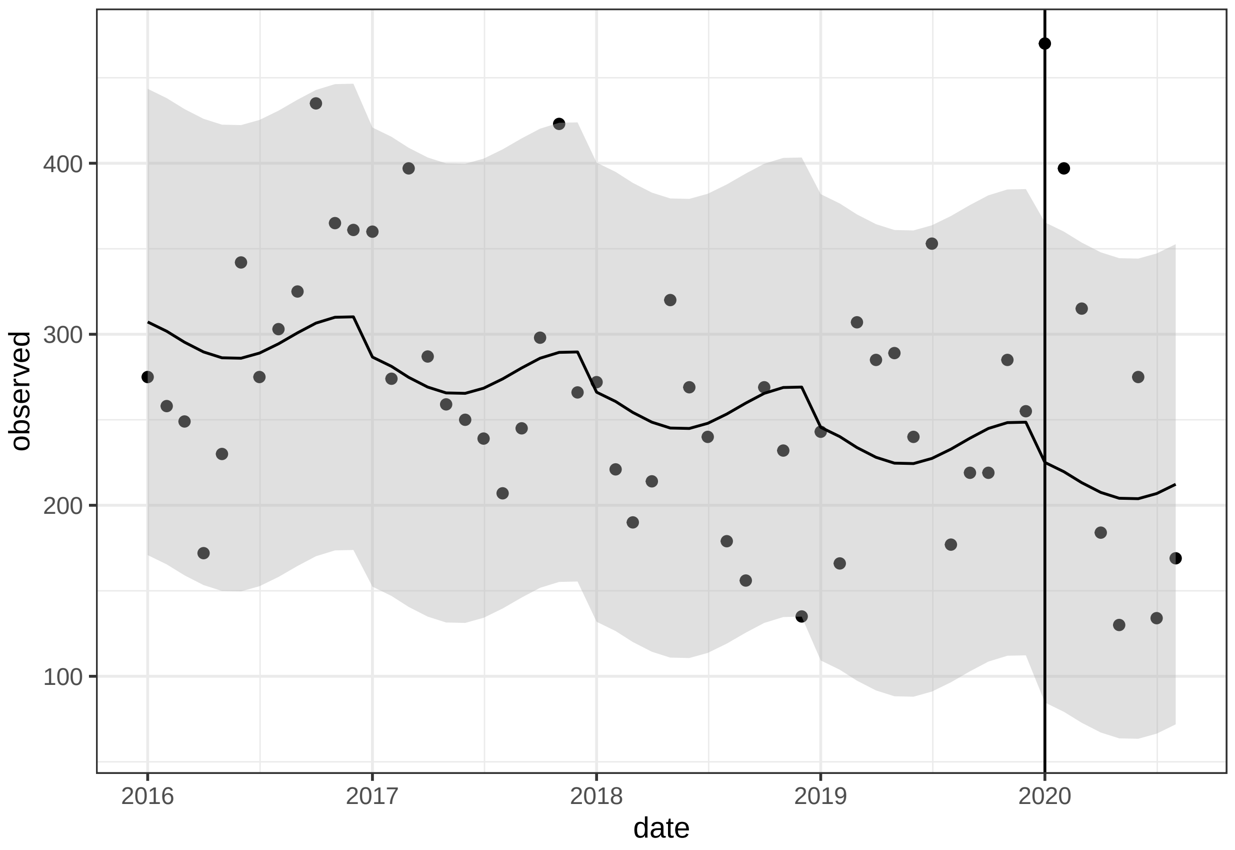
```
# Create 95% predicted values
predicted_values2 <- predict(fit2, facility_new, interval="predict")

predicted_values2 %>%
  data.frame() %>%
  mutate(date=facility_new$date,
         observed=facility_new$ari_count) -> predicted_values_new2

# deviations
predicted_values_new2 %>%
  mutate(deviations=observed-fit) %>%
  filter(date >= as.Date("2020-01-01"))
```

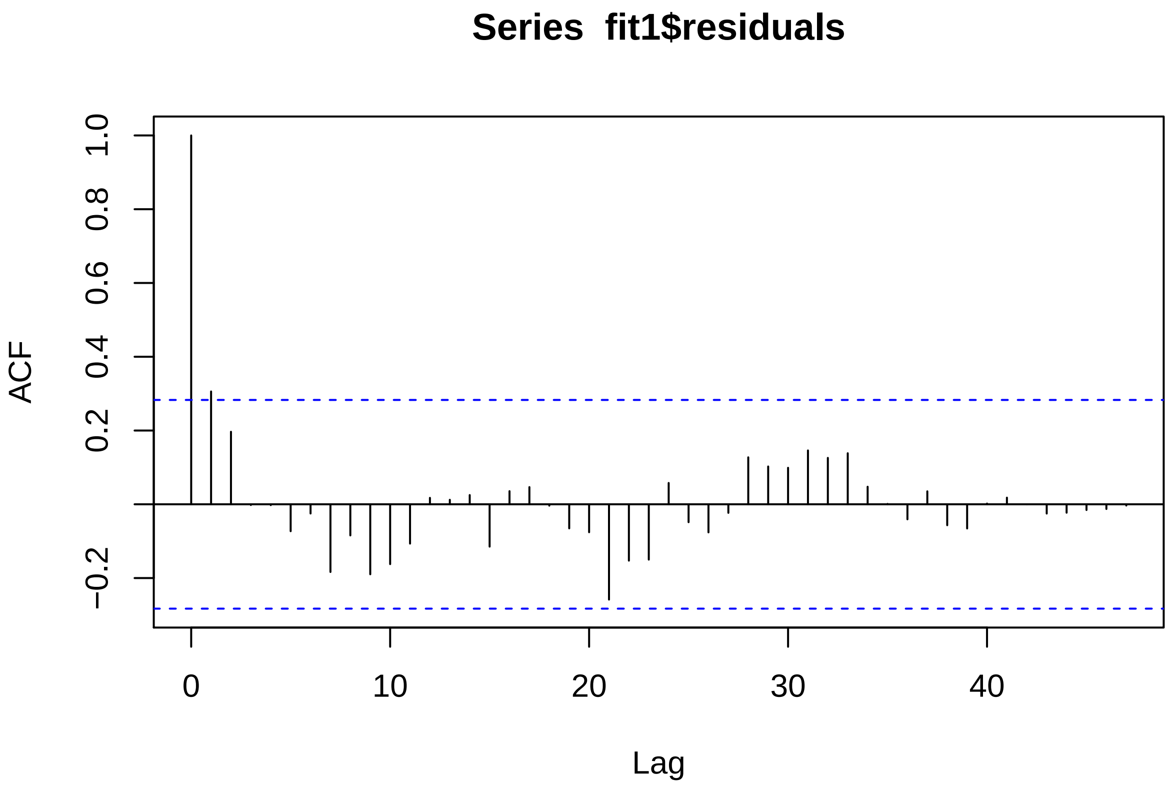
```
##         fit      lwr      upr       date observed deviations
## 49 225.0689 84.67435 365.4634 2020-01-01      470  244.93112
## 50 219.6338 79.23930 360.0284 2020-02-01      397  177.36616
## 51 213.2139 72.81941 353.6085 2020-03-01      315  101.78606
## 52 207.5294 67.13487 347.9239 2020-04-01      184  -23.52941
## 53 204.1034 63.70886 344.4979 2020-05-01      130  -74.10339
## 54 203.8539 63.45936 344.2484 2020-06-01      275   71.14610
## 55 206.8478 66.45324 347.2423 2020-07-01      134  -72.84778
## 56 212.2828 71.88829 352.6774 2020-08-01      169  -43.28283
```

```
ggplot(predicted_values_new2,aes(x=date,y=observed)) +
  geom_point() +
  geom_ribbon(aes(ymin=lwr,ymax=upr),fill = "grey70",alpha=.4) +
  geom_line(aes(x=date,y=fit)) +
  geom_vline(xintercept=as.Date("2020-01-01")) +
  theme_bw() +
  theme(legend.position="none")
```

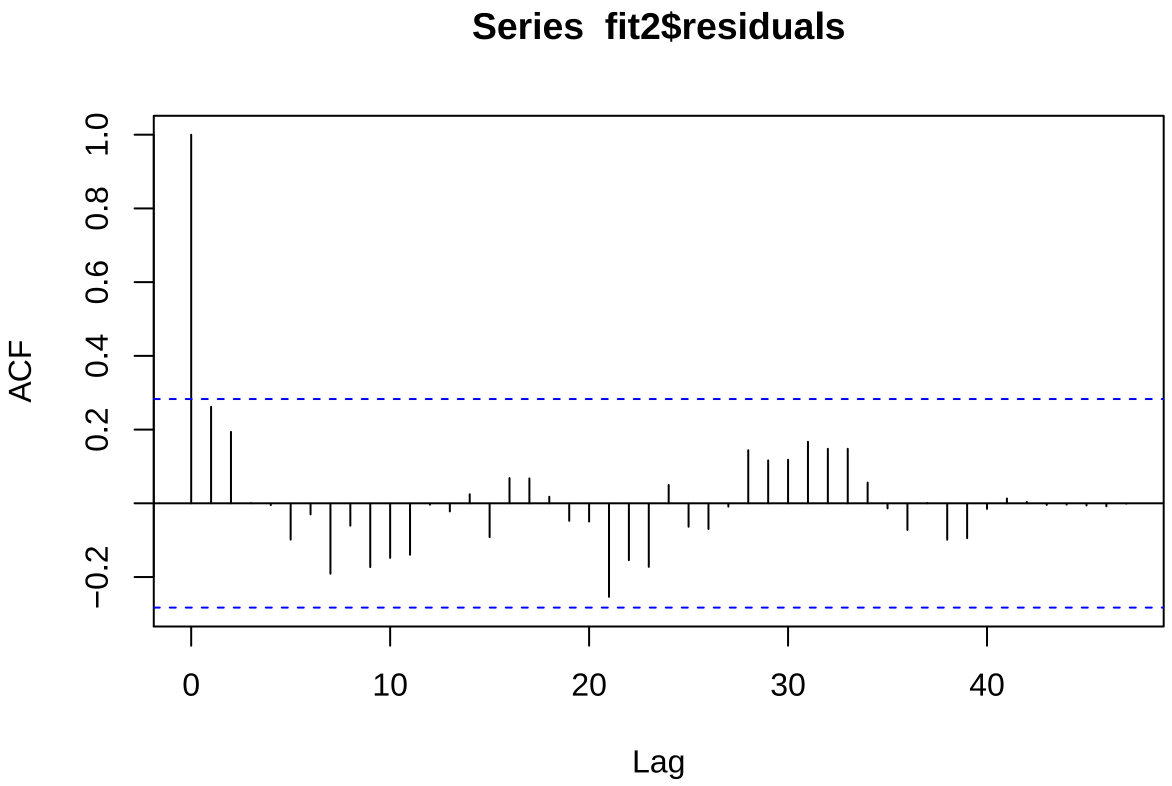


2. Compare the ACF plots between fit1 and fit2. Which looks better?

```
acf(fit1$residuals,lag.max=48)
```



```r
acf(fit2$residuals,lag.max=48)
```


Series fit2$residuals
ACF
Lag