Old Group level MHF COVID-19 analyses

Zachary Levine

2020-10-12

Methods

Basic questions:

- Seek the patients' perspectives around the importance of exercising during pandemic.
- Would also like to know whether they perceive their physicians to prioritize exercise and lifestyle behaviours any less importantly than they otherwise might, during this pandemic.

Variables and interactions of interest:

1. MET minutes
2. Volume of visits
3. Attendance
4. % Attendance and Met-minute interactions (because it is a sign of compliance)
5. Mean age and % females.

Time series methodology: Box-Jenkins

We're going to be folliwng the Box-Jenkins method for fitting autoregressive integrated moving average (ARIMA) models. This is a systematic and formulaic method of selecting, fitting, checking, ARIMA time series models. Our process is the following:

1. Model selection: Here, we ensure that our time series is stationary. We also need to identify seasonality, and seasonally difference if necessary. Then, we'll use plots of the autocorrelation (ACF) and partial autocorrelation (PACF) to decide if any autoregressive or moving average component should be used in the model, and to what (maximum) degree.
2. Next, we'll undertake parameter estimation using statistical software packages (in this case R). We'll preform some diagonstic plots to check our work.
3. In particular, the residuals should be independent of onenaother, and constant in variance and mean (of zero) over time. We'll display the mean and variance of residuals over time, in addition to Ljung–Box testing and ACF/PACF plots.
4. If the model is inadequate for our date, return to step one.

We want to compare pre-post trends around the week of march 16th within each year using p values. We also want to compare time trends in 2020 vs 2019 statistically.

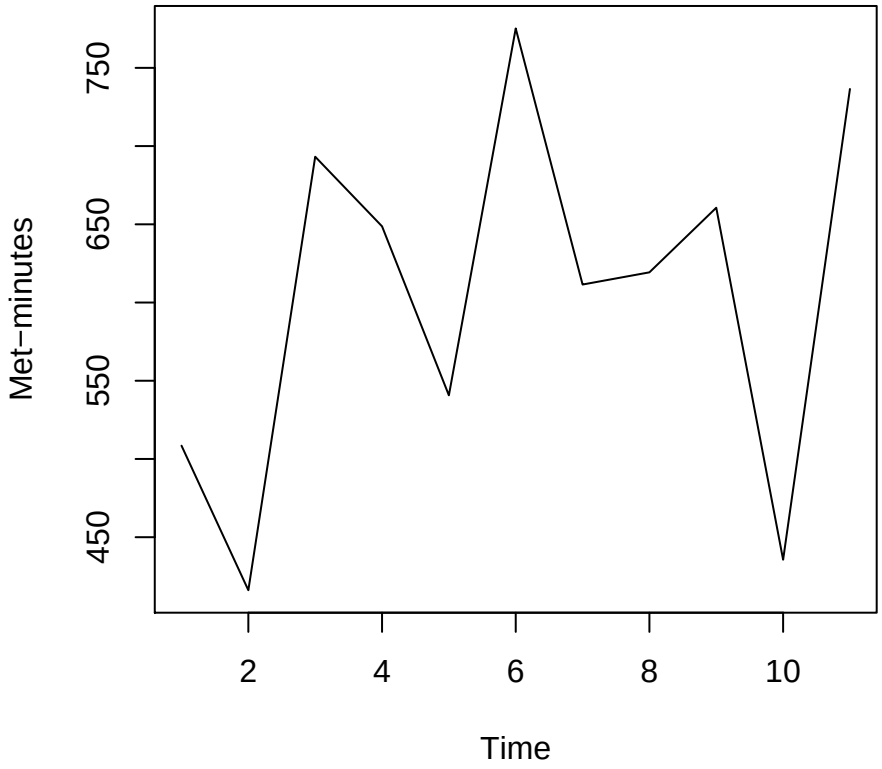
We also switch between piecewise regression (in comparing time trends) and dynamic regresion (structural change models) to account for the fact that the datapoints in 2019 and 2020 are not sequentiallyl linked in time, but weeks before and after the invervention are.

Comparing 2019 vs 2020 time trends

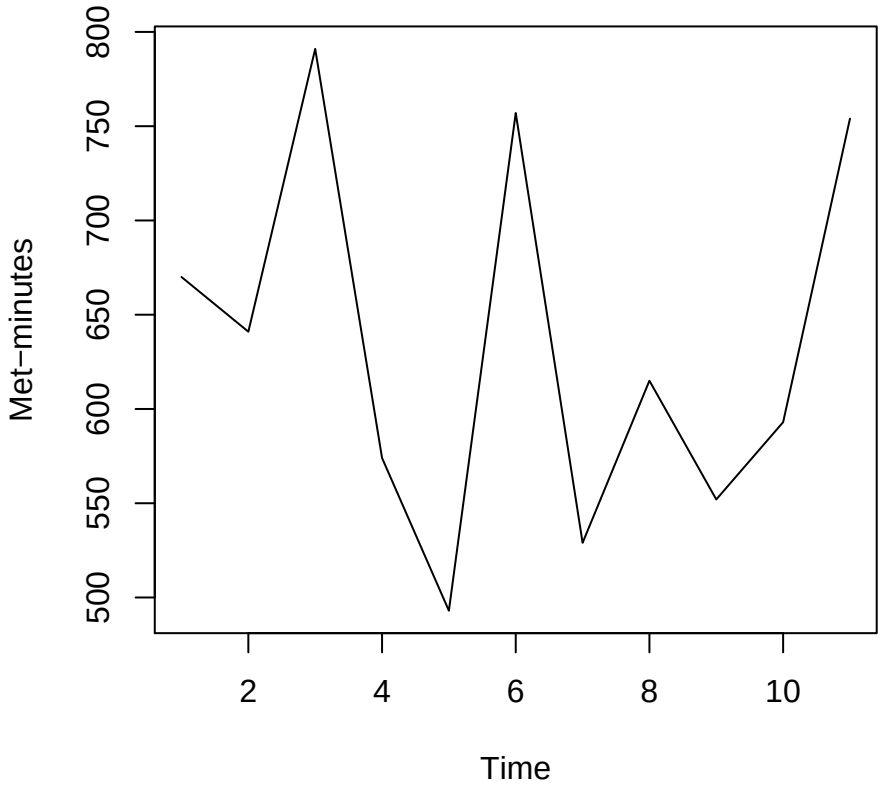
Met-minutes

We'll start by using `read_group_data()` to create the dataset we're going to use and properly format/type the attributes. This is more an exercise in data wrangling than anything else. We need to ascertain the model we're going to be using, and learn about the data and its features. The data runs from Jan 1 2019 until April 31 2019 and Jan 1 2019 until April 31, 2020. We begin by asking if the time series for 2019 and 2020 are stationary. Let's start with a visual inspection. We'll plot the variable against time. For *MET-minutes*, our plots for 2019 and 2020 are below.

```
df <- RMHF::read_group_data()
#> Warning: replacing previous import 'dplyr::intersect' by
#> 'lubridate::intersect' when loading 'RMHF'
#> Warning: replacing previous import 'dplyr::union' by 'lubridate::union'
#> when loading 'RMHF'
#> Warning: replacing previous import 'dplyr::setdiff' by 'lubridate::setdiff'
#> when loading 'RMHF'
df2019mets <- df[1:11, "Met-minutes"]
tsdf2019mets <- ts(df2019mets)
# Is the 2020 time-series stationary?
df2020mets <- df[12:22, "Met-minutes"]
tsdf2020mets <- ts(df2020mets)
# Met-minutes in 2019
plot(ts(RMHF::read_group_data()[1:11,"Met-minutes"]))
```




```
#  Met minutes in 2020
plot(ts(RMHF::read_group_data()[12:22,"Met-minutes"]))
```



We also note that there is no consistent trend in either time series over the entire time span. The series appear to wander up and down. There also don't appear to be any obvious outliers in either time series.

First, let's use two good (unit root) tests of stationarity (Augmented Dickey-Fuller, KPSS) to determine whether the time series is stationary. Let's start with the *Augmented-Dickey-Fuller* test. Here, the null hypothesis, *H0* is that the series is non-stationary, that is, the existence of a unit root. The alternative hypothesis is that the series is stationary. We'd like a p value less than 0.05 to reject the null.

```
df <- RMHF::read_group_data()
# Is the 2019 time-series stationary?
tseries::adf.test(tsdf2019mets, alternative = "stationary")
#>
#>  Augmented Dickey-Fuller Test
#>
#> data:  tsdf2019mets
#> Dickey-Fuller = -2.0025, Lag order = 2, p-value = 0.5714
#> alternative hypothesis: stationary
# Is the 2020 time-series stationary?
tseries::adf.test(tsdf2020mets, alternative = "stationary")
#>
#>  Augmented Dickey-Fuller Test
#>
#> data:  tsdf2020mets
#> Dickey-Fuller = -1.0793, Lag order = 2, p-value = 0.9089
#> alternative hypothesis: stationary
```

Alright, so the Augmented Dickey-Fuller test tells us we can't reject the null hypothesis that the series is non-stationary in either 2019 or 2020. This is important for us. Given that we have non-stationary data, we will need to "difference" the data until we obtain a stationary time series. We'll start with the "first-order"" difference. What we're doing here is that is removing the previous Y *met-minutes* values only once. For each

time point in our data, this gives you the change in value from the previous time point. So let's do this in parallel for 2019 and 2020.

```
# Differencing 2019 data and comparing the result.
arima(tsdf2019mets, order = c(0,0,0))
#>
#> Call:
#> arima(x = tsdf2019mets, order = c(0, 0, 0))
#>
#> Coefficients:
#>       intercept
#>        604.1727
#> s.e.    33.6618
#>
#> sigma^2 estimated as 12464:  log likelihood = -67.48,  aic = 138.95
tsdf2019mets_diff1<- diff(tsdf2019mets, differences = 1)
tseries::adf.test(tsdf2019mets_diff1, alternative = "stationary")
#>
#>  Augmented Dickey-Fuller Test
#>
#> data:  tsdf2019mets_diff1
#> Dickey-Fuller = -1.4182, Lag order = 2, p-value = 0.794
#> alternative hypothesis: stationary
arima(tsdf2019mets, order = c(0,1,0))
#>
#> Call:
#> arima(x = tsdf2019mets, order = c(0, 1, 0))
#>
#>
#> sigma^2 estimated as 32369:  log likelihood = -66.11,  aic = 134.23

# Differencing 2020 data and comparing the result.
arima(tsdf2020mets, order = c(0,0,0))
#>
#> Call:
#> arima(x = tsdf2020mets, order = c(0, 0, 0))
#>
#> Coefficients:
#>       intercept
#>        633.5455
#> s.e.    28.5741
#>
#> sigma^2 estimated as 8981:  log likelihood = -65.67,  aic = 135.35
tsdf2020mets_diff1 <- diff(tsdf2020mets, differences = 1)
tseries::adf.test(tsdf2020mets_diff1, alternative = "stationary")
#>
#>  Augmented Dickey-Fuller Test
#>
#> data:  tsdf2020mets_diff1
#> Dickey-Fuller = -1.9017, Lag order = 2, p-value = 0.6098
#> alternative hypothesis: stationary
arima(tsdf2020mets, order = c(0,1,0))
#>
#> Call:
```

```
#> arima(x = tsdf2020mets, order = c(0, 1, 0))
#>
#>
#> sigma^2 estimated as 23764:  log likelihood = -64.57,  aic = 131.14
```

Understanding our results Differencing the first time didn't make either time series stationary. But the standard deviation increased in both cases which indicates overdifferencing. In addition, the kpss test shows that both series were stationary from the get-go.

```
tseries::kpss.test(tsdf2019mets)
#> Warning in tseries::kpss.test(tsdf2019mets): p-value greater than printed
#> p-value
#>
#>  KPSS Test for Level Stationarity
#>
#> data:  tsdf2019mets
#> KPSS Level = 0.24367, Truncation lag parameter = 2, p-value = 0.1
tseries::kpss.test(tsdf2020mets)
#> Warning in tseries::kpss.test(tsdf2020mets): p-value greater than printed
#> p-value
#>
#>  KPSS Test for Level Stationarity
#>
#> data:  tsdf2020mets
#> KPSS Level = 0.17184, Truncation lag parameter = 2, p-value = 0.1
```

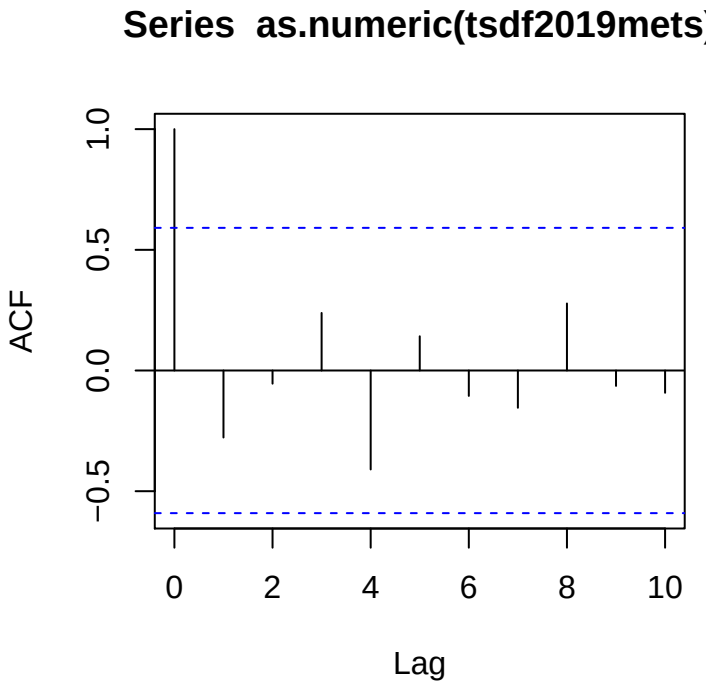
Because differencing appears to worsen the model fit in the best case, and in the worst case we don't need it, let's assume the time series are stationary. auto.arima in R uses the results from the kpss test over those from the adf test by default anyway. So let's assume they are.

Now we should check if the time series are seasonal. Let's check for patterns or regularity in the autocorrelograms and partialcorrelograms. If there is a pattern are, we should extract its frequency from both time series. As a quick refresher, p refers to how many previous (lagged) Y values are accounted for at each point, and q refers to how many previous (lagged) *error* values are accounted for each time point in our model.

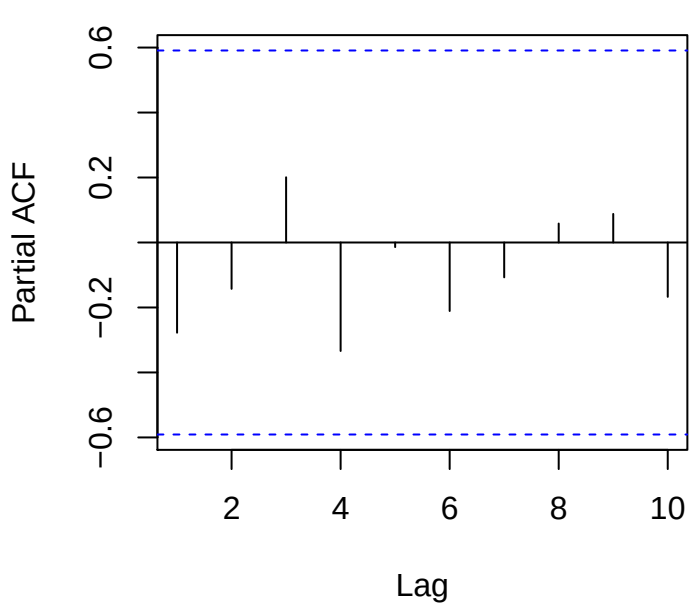
A correlogram plot of the correlation of each MET-minute variable at time t with that at time t-k. A partial correlogram the same except for the fact that it removes the effect of shorter autocorrelation lags when calculating the correlation at longer lags. Techhnically speaking, the partial correlation at lag k is the autocorrelation between Yt and Yt-k that is NOT accounted for by the autocorrelations from the 1st to the (k-1)st lags.

Let's explore the correlogram and partial correlograms for the 2019 and 2020 data, respectively.

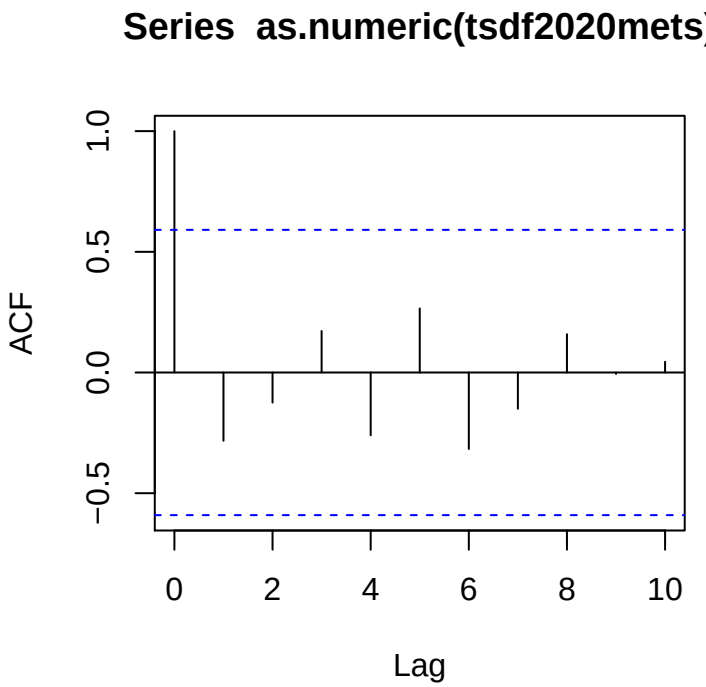
```
acf(as.numeric(tsdf2019mets))
```



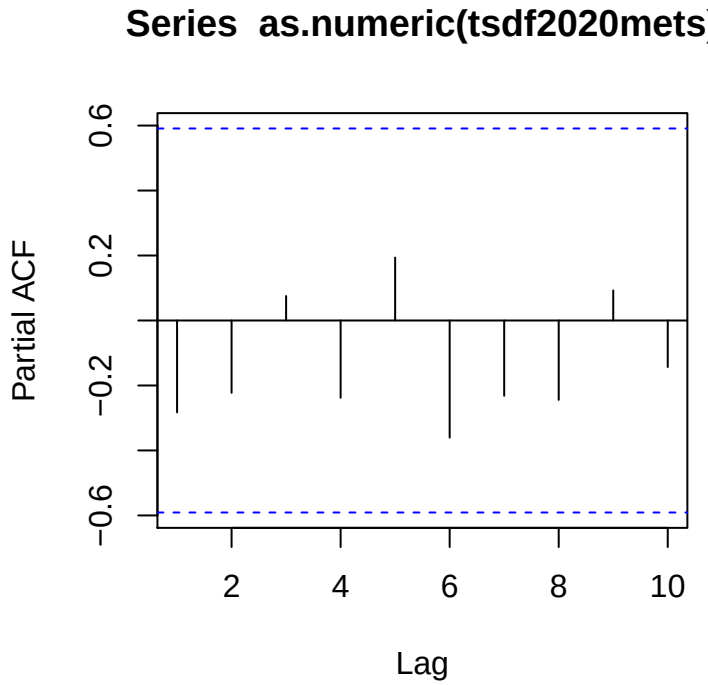

```
pacf(as.numeric(tsdf2019mets))
```




```
acf(as.numeric(tsdf2020mets))
```




```
pacf(as.numeric(tsdf2020mets))
```



The 0th lag will always be significant but this tells us nothing. What we do see is a lack of a pattern in both the acf and pacf for both plots. There's just statistically insignificant randomness. This points to a lack of seasonality. Indeed, if we try and extract the frequency from both objects. we obtain one, indicating no seasonality.

```
#Extract frequency from 2019 data.
forecast::findfrequency(tsdf2019mets)
#> [1] 1
#Extract frequency from 2020 data.
```

```
forecast::findfrequency(tsdf2020mets)
#> [1] 1
```

This agrees with *"Wiener–Khinchin theorem, also known as the Wiener–Khintchine theorem and sometimes as the Wiener–Khinchin–Einstein theorem or the Khinchin–Kolmogorov theorem, [which] states that the autocorrelation function of a wide-sense-stationary random process has a spectral decomposition given by the power spectrum of that process"* (Wikipedia).

But what if we were to decompose each year by seasonal periods of 2, 3, 4, or 5 weeks (the maximum for a series of 11 points to be periodic)? Seastest does exactly this by using an F test on seasonal dummies and checking for statistically significant seasonality coefficients (betas). Essentially, we fit the data to a regression model encoded with seasonal dummy variables and we test each parameter estimate for statistical significance at the 0.05 level.

```
testSeasonality <- function(listobj){
  possibleFreqs <- 2:5
  for (thefreq in possibleFreqs){
    print(paste0("Is seasonal at a frequency of: ", thefreq, " ?" ))
    print(seastests::isSeasonal(ts(listobj), freq = thefreq, test = "seasdum"))
  }
}
#Try 2019.
testSeasonality(df2019mets)
#> [1] "Is seasonal at a frequency of: 2 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 3 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 4 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 5 ?"
#> [1] FALSE
#Try 2020.
testSeasonality(df2020mets)
#> [1] "Is seasonal at a frequency of: 2 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 3 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 4 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 5 ?"
#> [1] FALSE
```

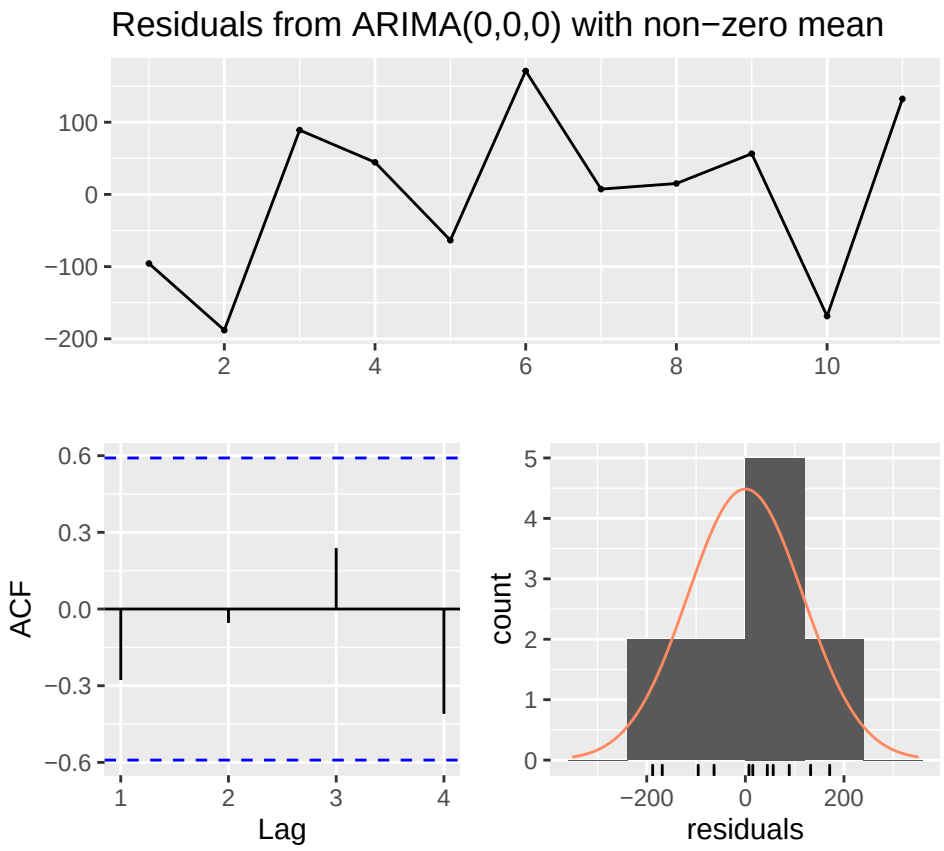
We've now concluded that neither time series is seasonal, and that both are stationary (to the extent that differencing worsens the model fit, which auto.arima() also agrees with). Now that we have a stationary time series for each year, we should fit the appropriate nonseasonal ARIMA model. This means finding the most appropriate values for p and q, as $d = 0$, for both models.

The largest statistically significant lag values of the correlogram gives the possible q values for the ARIMA model. In addition, the largest statistically significant lag of the partial correlogram gives the p value for an ARIMA model. Looking back at our ACF and PACFs above, we note that our time series appears to fit best a *white noise* model. There are no autoregressive or moving average terms required here. In other words, observations at each time point are statistically independent of oneanother. So we're left with ARIMA(0,0,0) models for both years, which *auto.arima* agrees with.

Checking and fitting ARIMA(0,0,0) models:

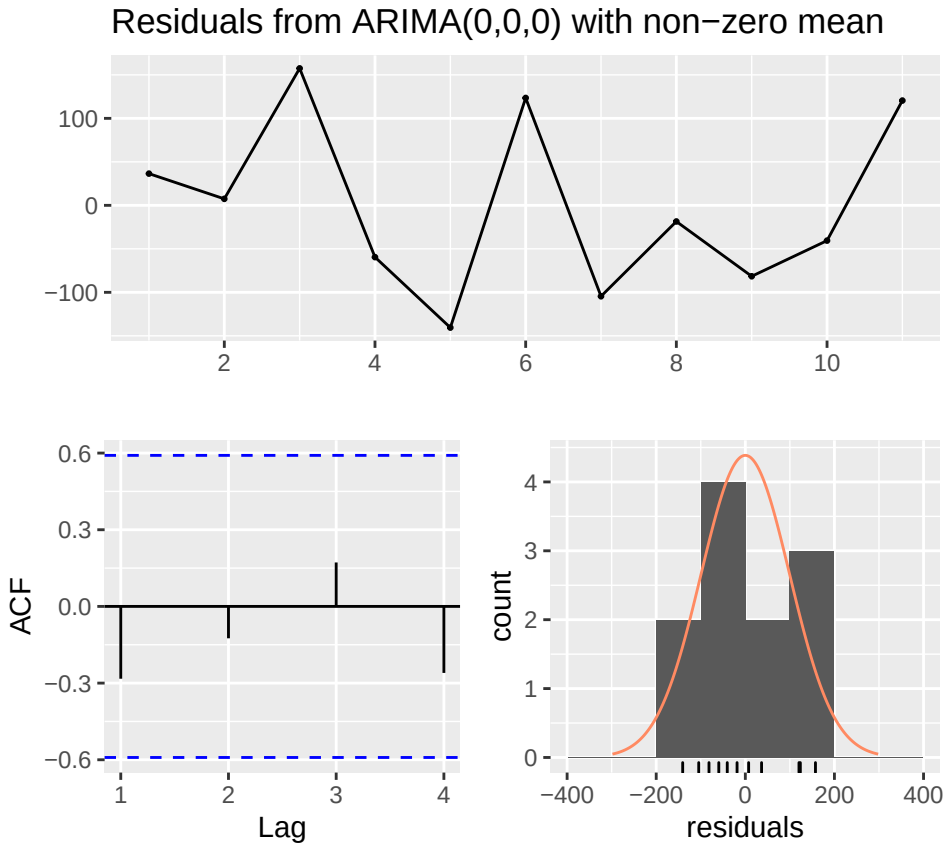
```
#Fit both ARIMA models.
arima2019 <- forecast::Arima(tsdf2019mets)
arima2020 <- forecast::Arima(tsdf2020mets)


#Check 2019 ARIMA.
forecast::checkresiduals(arima2019)
```




```
#>
#>  Ljung-Box test
#>
#> data:  Residuals from ARIMA(0,0,0) with non-zero mean
#> Q* = 5.5992, df = 3, p-value = 0.1328
#>
#> Model df: 1.   Total lags used: 4

#Check 2020 ARIMA.
forecast::checkresiduals(arima2020)
```




```
#>
#>  Ljung-Box test
#>
#> data:  Residuals from ARIMA(0,0,0) with non-zero mean
#> Q* = 3.3024, df = 3, p-value = 0.3473
#>
#> Model df: 1.   Total lags used: 4
```

The Ljung-Box statistic tests the null hypothesis that the residuals are distributed independently *(H0)*, vs the *(Ha)* hypothesis that the residuals exhbit some form of autocorrelation (the model shows lack of fit). In other words, if the p value is greater than 0.05 then the residuals are independent. We want $p > 0.05$, which we have in both cases.

ARIMA Conclusions Since, our data is white noise we're not going to be able to forecast using classical time-series models. This makes our original plan for intervention analyses unachievable unless we switch models. We have no statistically significnat autocorrelation here though.

What we can do is test whether the linear time trend differs between 2019 and 2020.

```
df <- RMHF::read_group_data()
colnames(df) <- c(colnames(df)[1:10], "mets")
df$mets <- as.numeric(df$mets)
df$weeks <- c(1:11, 1:11)
#Create a dummy variable for the year.
df$year <- c(rep(0, 11), rep(1, 11))
```

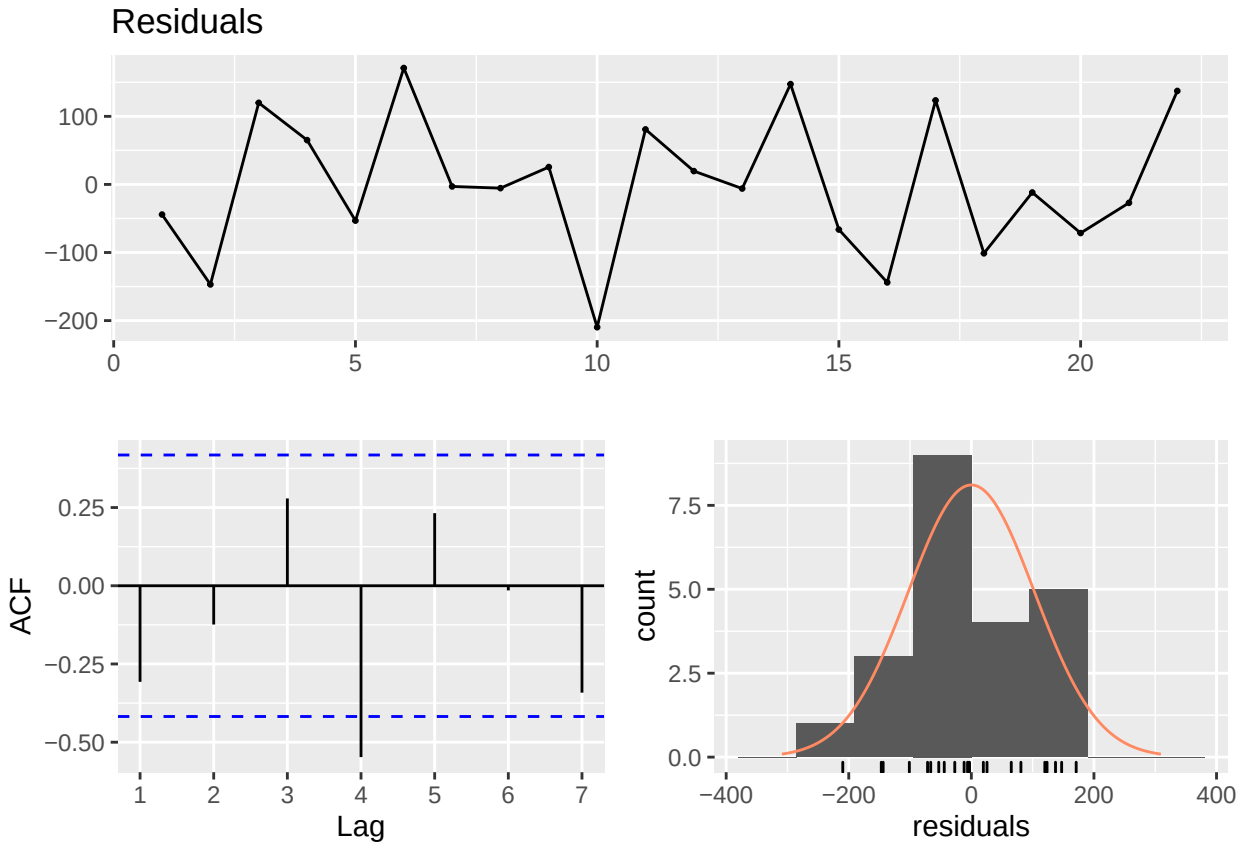
Methodology What we're going to do here is include an interaction term between year (2019, 2020) and time, whose coefficient (trend) we want to compare. Since we think that time might have a different effect for 2019 and 2020, we include an interaction term between year and each time(weeks). Then, we fit a regression

model to the data. The coefficient for weeks is the coefficient for weeks for the reference group (2019) only. The interaction term between year and weeks represents the difference in the coefficients between 2019 and 2020. Lastly, to obtain the coefficient for 2020 (the comparison group), all we need to to is add the coefficients for the predictor (weeks) on its own and week*year interaction. This will give us a p value, which is good. So let's fit the regression line.

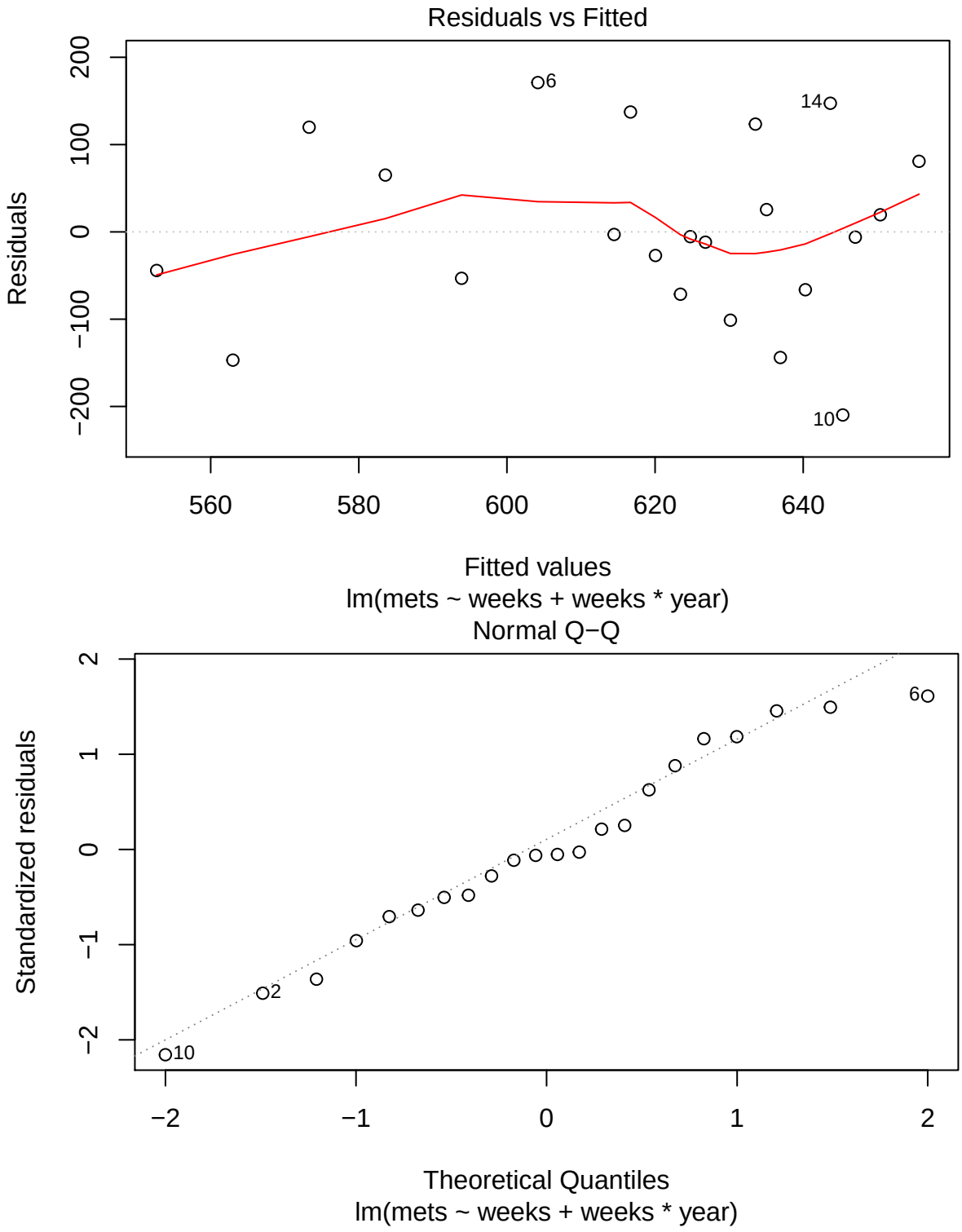
```
#Fit the regression.
fit <- lm(mets ~ weeks + weeks*year, data  = df)
```

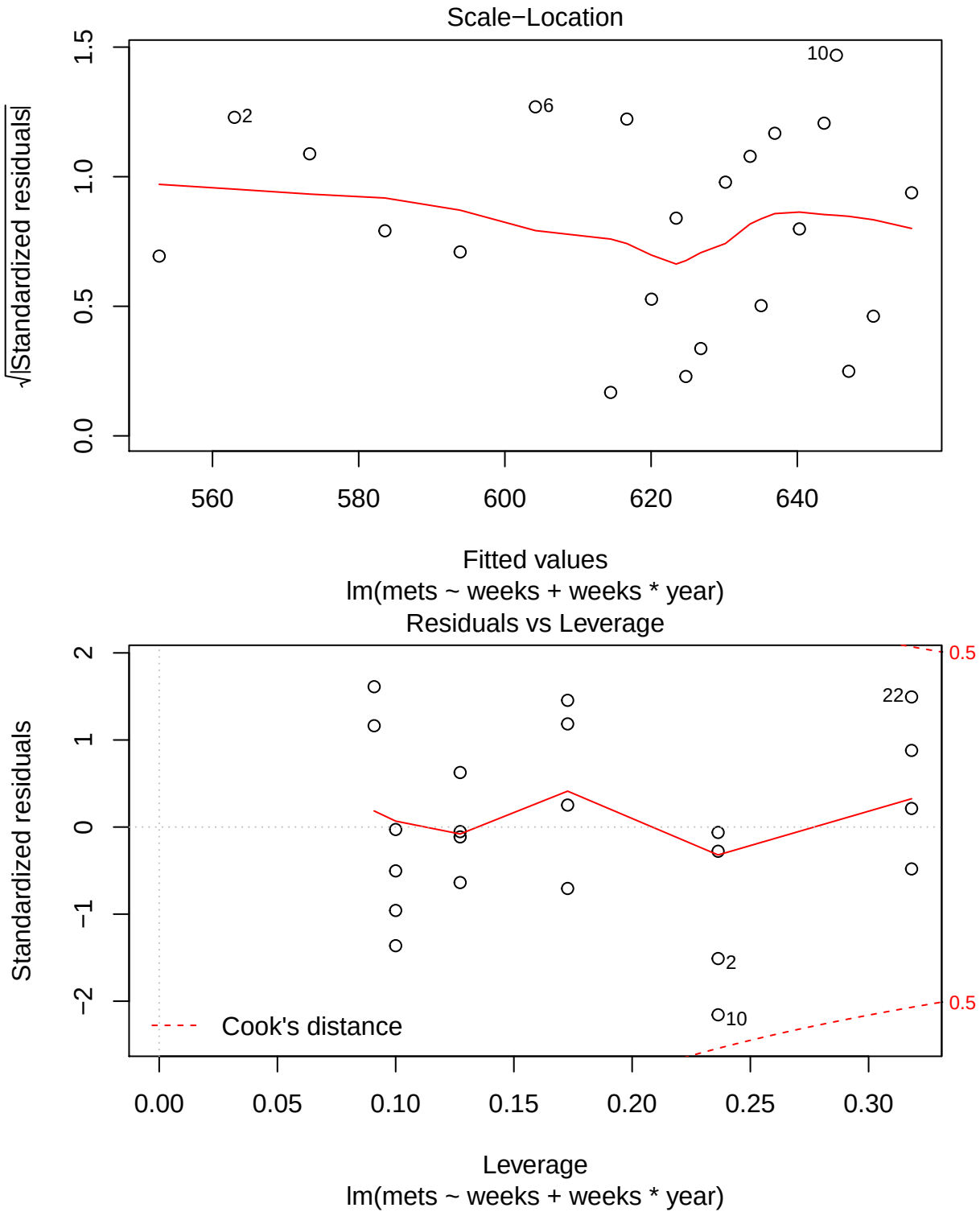
Let's check the assumptions of our model using a few diagnostic plots.

```
forecast::checkresiduals(fit)
```




```
#>
#>  Breusch-Godfrey test for serial correlation of order up to 7
#>
#> data:  Residuals
#> LM test = 15.914, df = 7, p-value = 0.02592
plot(fit)
```


Residuals vs Fitted
Residuals
200
100
0
−100
−200
6
14
10
560
580
600
620
640
Fitted values
lm(mets ~ weeks + weeks * year)
Normal Q−Q
Standardized residuals
2
1
0
−1
−2
6
2
10
−2
−1
0
1
2
Theoretical Quantiles
lm(mets ~ weeks + weeks * year)



A quick refresher that we don't want to reject the null in the Breusch-Godfrey test, as it's that there's no serial correlation in our data. In other words, we want $p > 0.05$ for this to work. Checking the other diagnostic plots tells us that our linear models work as well. And let's grab the p value for each interaction term.

```
anova(fit)
#> Analysis of Variance Table
```

```
#>
#> Response: mets
#>            Df Sum Sq Mean Sq F value Pr(>F)
#> weeks       1   2634  2633.8  0.2126 0.6503
#> year        1   4745  4745.2  0.3830 0.5437
#> weeks:year  1  10271 10271.0  0.8291 0.3746
#> Residuals  18 222994 12388.6
summary(fit)
#>
#> Call:
#> lm(formula = mets ~ weeks + weeks * year, data = df)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -209.744  -63.013   -5.747   76.926  171.027
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   542.42      71.98   7.536 5.68e-07 ***
#> weeks          10.29      10.61   0.970    0.345
#> year          111.37     101.79   1.094    0.288
#> weeks:year    -13.67      15.01  -0.911    0.375
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 111.3 on 18 degrees of freedom
#> Multiple R-squared:  0.07334,    Adjusted R-squared:  -0.0811
#> F-statistic: 0.4749 on 3 and 18 DF,  p-value: 0.7036
```

Some preliminary conclusions From the p value for the interaction term > 0.05, we can conclude that linear time trends for met-minutes do not differ significantly between 2020 and 2019.

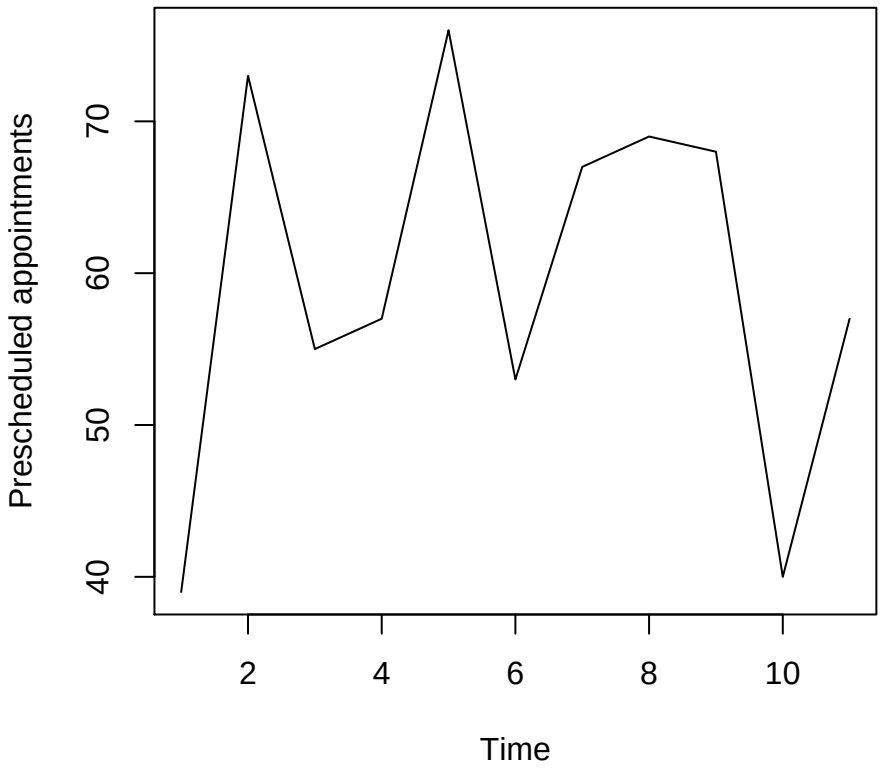
Volume of visits

We'll start by using `read_group_data()` to create the dataset we're going to use and properly format/type the attributes. We need to ascertain the model we're going to be using, and learn about the data and its features. We begin by asking if the time series for 2019 and 2020 are stationary. Let's start with a visual inspection. We'll plot the variable against time. For *Volume of visits*, our plots for 2019 and 2020 are below.

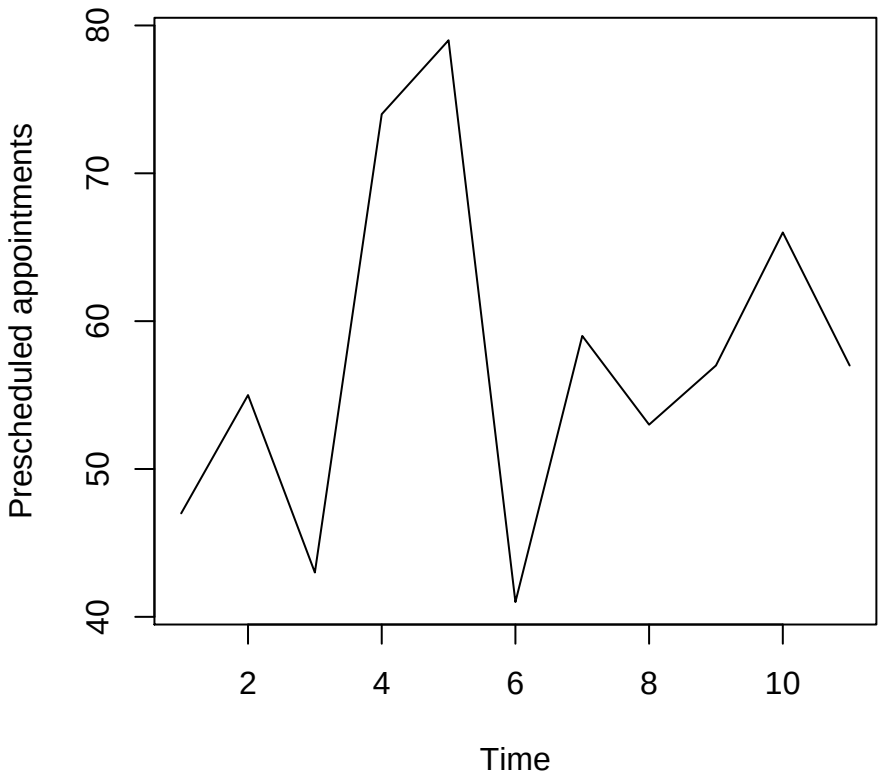
```
df <- RMHF::read_group_data()
df2019vol <- df[1:11, "Prescheduled appointments" ]
ts2019vol <- ts(df2019vol)

df2020vol <- df[12:22, "Prescheduled appointments" ]
ts2020vol <- ts(df2020vol)

plot(ts2019vol)
```




```
plot(ts2020vol)
```



We also note that there is no consistent trend in either time series over the entire time span. The series appear to wander up and down. There also don't appear to be any obvious outliers in either time series.

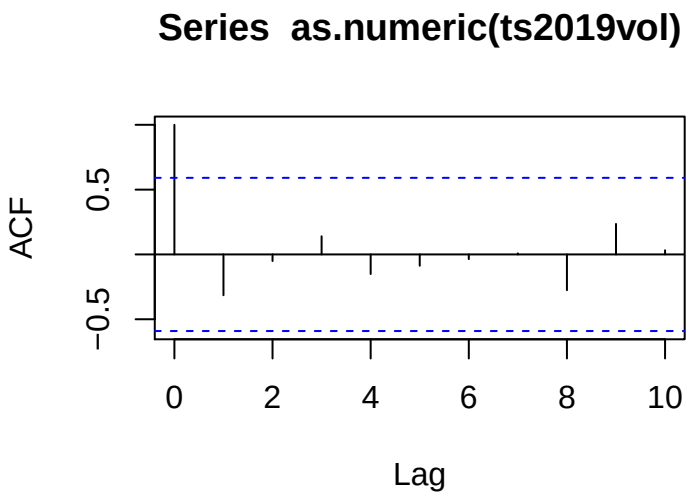
We'l start with a KPSS unit root test on both time series.

```
tseries::kpss.test(ts2019vol)
#> Warning in tseries::kpss.test(ts2019vol): p-value greater than printed p-
#> value
#>
#>  KPSS Test for Level Stationarity
#>
#> data:  ts2019vol
#> KPSS Level = 0.15974, Truncation lag parameter = 2, p-value = 0.1
tseries::kpss.test(ts2020vol)
#> Warning in tseries::kpss.test(ts2020vol): p-value greater than printed p-
#> value
#>
#>  KPSS Test for Level Stationarity
#>
#> data:  ts2020vol
#> KPSS Level = 0.14641, Truncation lag parameter = 2, p-value = 0.1
```

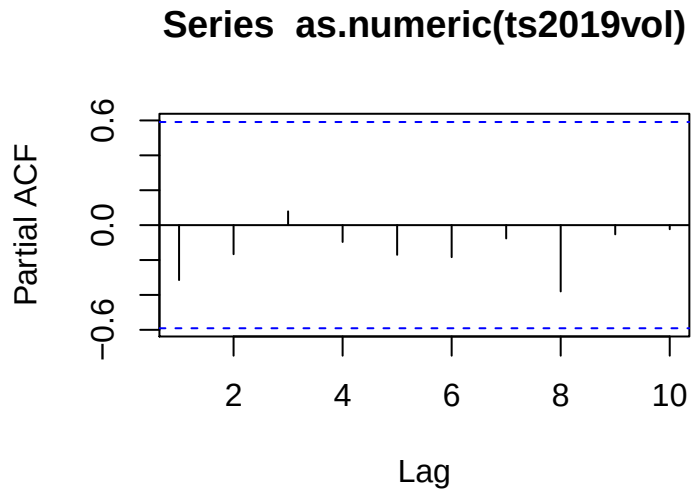
The KPSS test shows that both series are stationary from the get-go.

Now we should check if the time series are seasonal. Let's check for patterns or regularity in the autocorrelograms and partialcorrelograms.

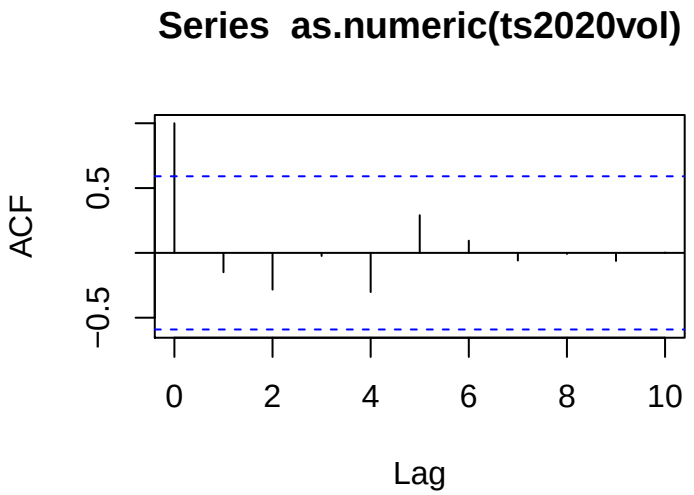
```
acf(as.numeric(ts2019vol))
```



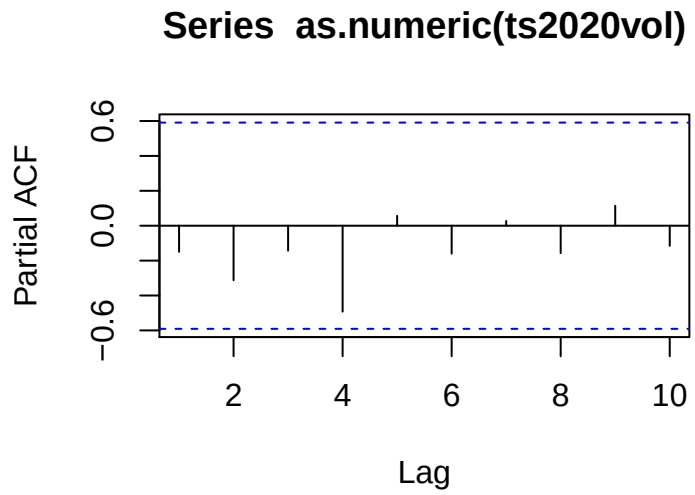

```
pacf(as.numeric(ts2019vol))
```




```
acf(as.numeric(ts2020vol))
```




```
pacf(as.numeric(ts2020vol))
```



No lags appear to be statistically significant on either the correlogram or partial correlograms for each year. Fourier analysis below:

```
#Extract frequency from 2019 data.
forecast::findfrequency(ts2019vol)
#> [1] 1
#Extract frequency from 2020 data.
forecast::findfrequency(ts2020vol)
#> [1] 1
```

This agrees with *"Wiener–Khinchin theorem,* as we'd like (Wikipedia).

But what if we were to decompose each year by seasonal periods of 2, 3, 4, or 5 weeks (the maximum for a series of 11 points to be periodic)? Seastest does exactly this by using an F test on seasonal dummies and checking for statistically significant seasonality coefficients (betas).

```
#Try 2019.
testSeasonality(ts2019vol)
#> [1] "Is seasonal at a frequency of: 2 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 3 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 4 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 5 ?"
#> [1] FALSE
#Try 2020.
testSeasonality(ts2020vol)
#> [1] "Is seasonal at a frequency of: 2 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 3 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 4 ?"
#> [1] FALSE
#> [1] "Is seasonal at a frequency of: 5 ?"
#> [1] FALSE
```

We've now concluded that neither time series is seasonal, and that both are stationary. Looking back at our ACF and PACFs above, we note that our time series appears to fit best a *white noise* model. There are no autoregressive or moving average terms required here. In other words, observations at each time point are statistically independent of oneanother. So we're left with ARIMA(0,0,0) models for both years, which *auto.arima* agrees with.

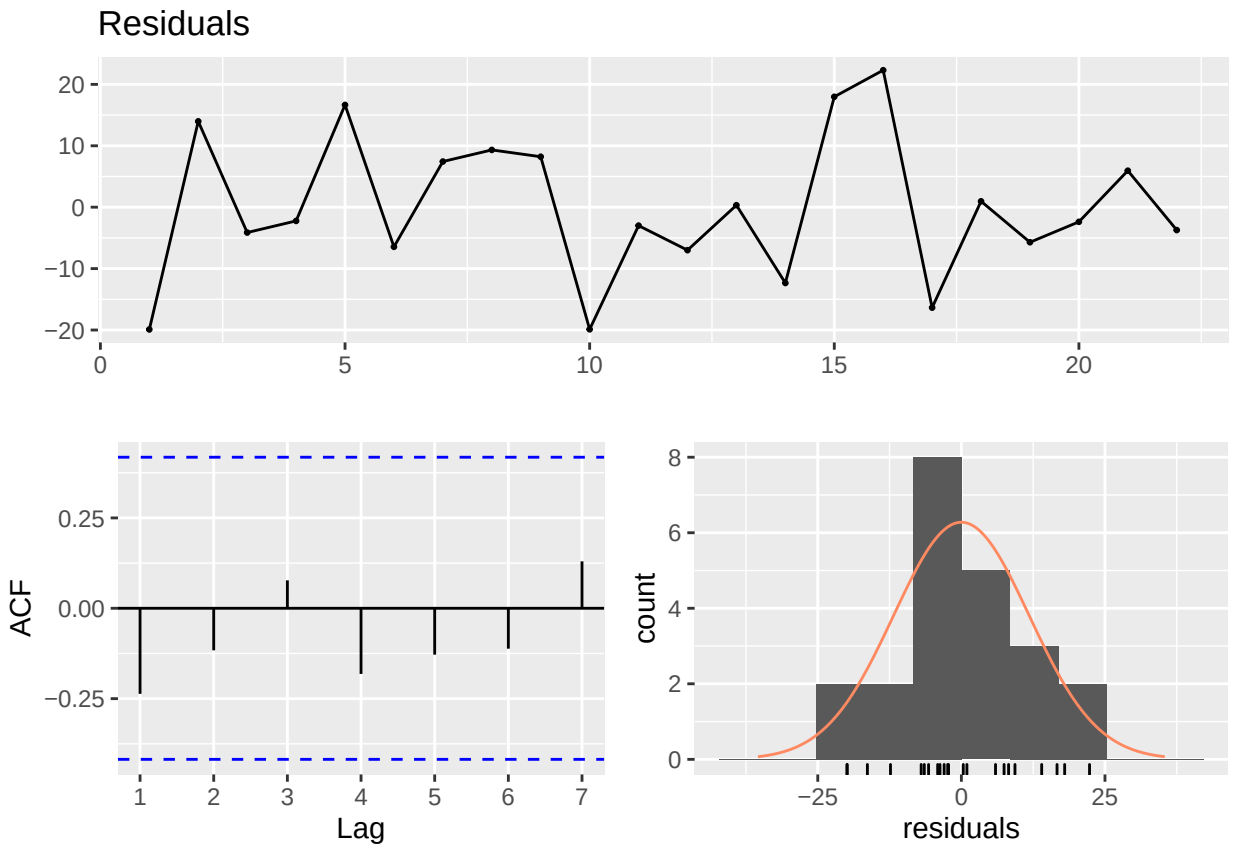
ARIMA Conclusions

Once again, we're left with white noise. So let's fit the regression line model we'll be using. We can test whether the linear time trend differs between 2019 and 2020.

```
df <- RMHF::read_group_data()
colnames(df) <- c(colnames(df)[1:2], "vol", colnames(df)[4:length(colnames(df))])
df$vol <- as.numeric(df$vol)
df$weeks <- c(1:11, 1:11)
#Create a dummy variable for the year.
df$year <- c(rep(0, 11), rep(1, 11))
#Fit the regression.
fitvol <- lm(vol ~ weeks + weeks*year, data  = df)
```

Let's check the assumptions of our model using a few diagnostic plots.

```
forecast::checkresiduals(fitvol)
```




```
#>
#>  Breusch-Godfrey test for serial correlation of order up to 7
#>
#> data:  Residuals
#> LM test = 13.135, df = 7, p-value = 0.06888
anova(fit)
#> Analysis of Variance Table
#>
#> Response: mets
#>             Df Sum Sq Mean Sq F value Pr(>F)
#> weeks        1   2634  2633.8  0.2126 0.6503
#> year         1   4745  4745.2  0.3830 0.5437
#> weeks:year   1  10271 10271.0  0.8291 0.3746
#> Residuals   18 222994 12388.6
summary(fitvol)
#>
#> Call:
#> lm(formula = vol ~ weeks + weeks * year, data = df)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -19.909  -6.268  -2.309   8.023  22.309
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
```

```
#> (Intercept)  58.8000     8.2451   7.131 1.21e-06 ***
#> weeks         0.1091     1.2157   0.090    0.929
#> year         -5.4727    11.6604  -0.469    0.644
#> weeks:year    0.5636     1.7192   0.328    0.747
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 12.75 on 18 degrees of freedom
#> Multiple R-squared:  0.02503,    Adjusted R-squared:  -0.1375
#> F-statistic: 0.1541 on 3 and 18 DF,  p-value: 0.9257
```

Thoughts and conclusions

Difference again not significant.

COVID-19 Intervention Analyses

We're going to be using dynamic regression here.

MET minutes

If we fit an ARIMA model to the 2020 data automatically, we yield the following automatic model fit.

```
autofit <- forecast::auto.arima(ts(RMHF::read_group_data()[12:22, "Met-minutes"]))
```

This tells us the ideal ARIMA model is a white noise model. So we can use simple linear regression for an algebraically equivalent model.

```
theDf <- data.frame("y" = as.numeric(RMHF::read_group_data()[12:22,]$"Met-minutes"), "weeks" = 1:11,  "
fullLm <- lm(data = theDf, formula = y ~ weeks + weeks*int)
half <- lm(data = theDf, formula = y ~ weeks)
```

Now, if we use the likelihood ratio test of nested models, we can get the p value associated with the effect of the intervention.

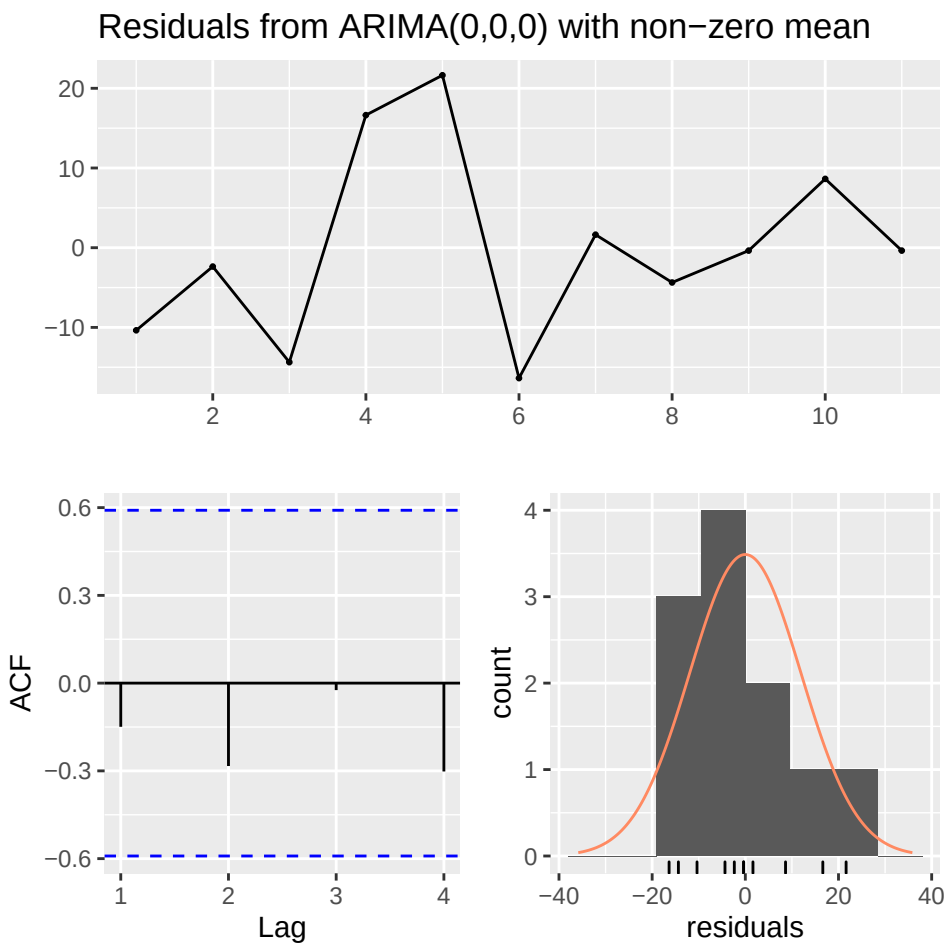
```
lmtest::lrtest(fullLm, half)
#> Likelihood ratio test
#>
#> Model 1: y ~ weeks + weeks * int
#> Model 2: y ~ weeks
#>   #Df  LogLik Df  Chisq Pr(>Chisq)
#> 1   5 -64.088
#> 2   3 -65.604 -2 3.0323     0.2196
```

This test compares the goodness of fit of two nested models (which we have). The null hypothesis in this test is that the smaller model fits the data better. So, if we reject *H0*, then the larger model is statistically significantly better than the smaller one. In this case, the larger model which also regresses on the presence of the intervention, is not statistically significantly better for our data. What does this tell us? The presence of the intervention did not change met minutes too much.

Volume of visits

If we fit an ARIMA model to the 2020 data automatically, we yield the following automatic model fit.

```
autofit <- forecast::auto.arima(ts(RMHF::read_group_data()[12:22, "Prescheduled appointments"]))
forecast::checkresiduals(autofit)
```




```
#>
#>  Ljung-Box test
#>
#> data:  Residuals from ARIMA(0,0,0) with non-zero mean
#> Q* = 3.4679, df = 3, p-value = 0.3249
#>
#> Model df: 1.   Total lags used: 4
```

This tells us the ideal ARIMA model is a white noise model. So let's use simple linear regression here (which our ARIMA is algebraically equivalent to, anyway)

```
theDf <- data.frame("y" = as.numeric(RMHF::read_group_data()[12:22,]$"Prescheduled appointments"), "wee
fullLm <- lm(data = theDf, formula = y ~ weeks + weeks*int)
half <- lm(data = theDf, formula = y ~ weeks)
```

Now, if we use the likelihood ratio test of nested models, we can get the p value associated with the effect of the intervention.

```
lmtest::lrtest(fullLm, half)
#> Likelihood ratio test
#>
#> Model 1: y ~ weeks + weeks * int
#> Model 2: y ~ weeks
```

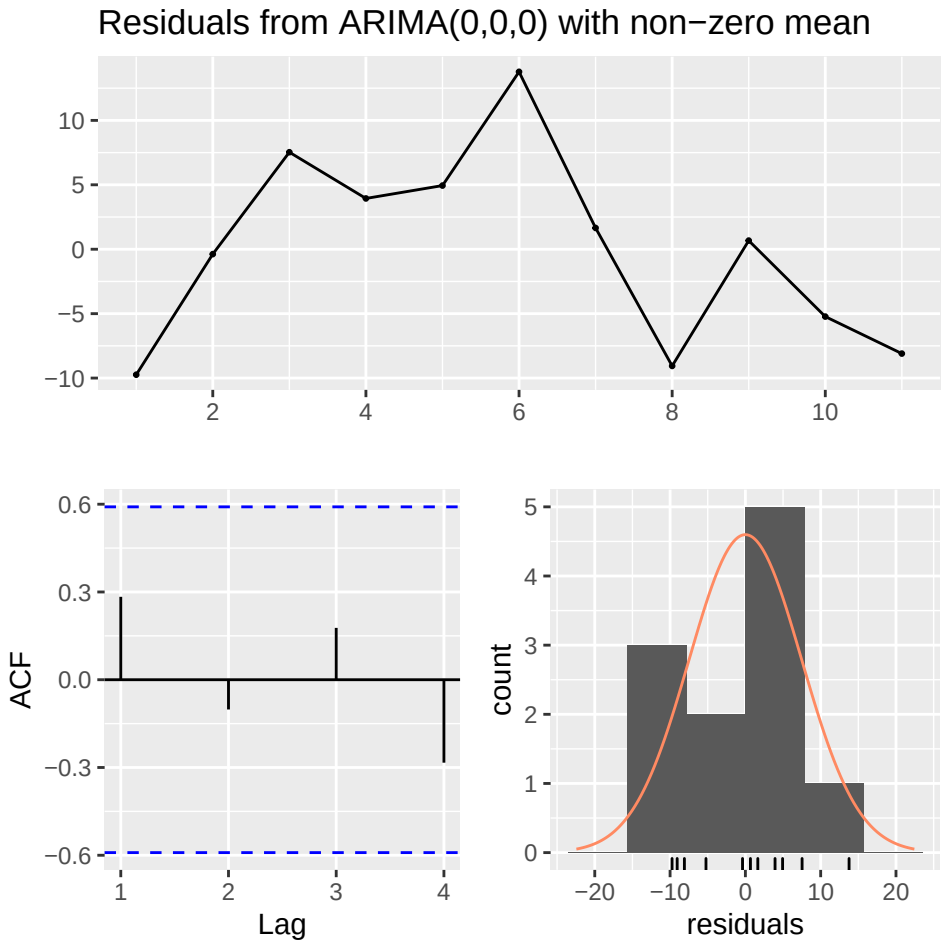
```
#>   #Df  LogLik Df  Chisq Pr(>Chisq)
#> 1   5 -41.999
#> 2   3 -42.179 -2 0.3592     0.8356
```

In this case, the larger model which also regresses on the presence of the intervention, is not statistically significantly better for our data. What does this tell us? The presence of the intervention did not volume of visits minutes too much.

Attendance

Automatic model fit.

```
autofit <- forecast::auto.arima(ts(RMHF::read_group_data()[12:22, "% of patients who were no-shows"]))
forecast::checkresiduals(autofit)
```




```
#>
#>  Ljung-Box test
#>
#> data:  Residuals from ARIMA(0,0,0) with non-zero mean
#> Q* = 3.5181, df = 3, p-value = 0.3184
#>
#> Model df: 1.   Total lags used: 4
```

This tells us the ideal ARIMA model is a white noise model. So let's use simple linear regression here.

```
theDf <- data.frame("y" = as.numeric(RMHF::read_group_data()[12:22,]$"% of patients who were no-shows")
fullLm <- lm(data = theDf, formula = y ~ weeks + weeks*int)
half <- lm(data = theDf, formula = y ~ weeks)
```

Now, if we use the likelihood ratio test of nested models, we can get the p value associated with the effect of the intervention.

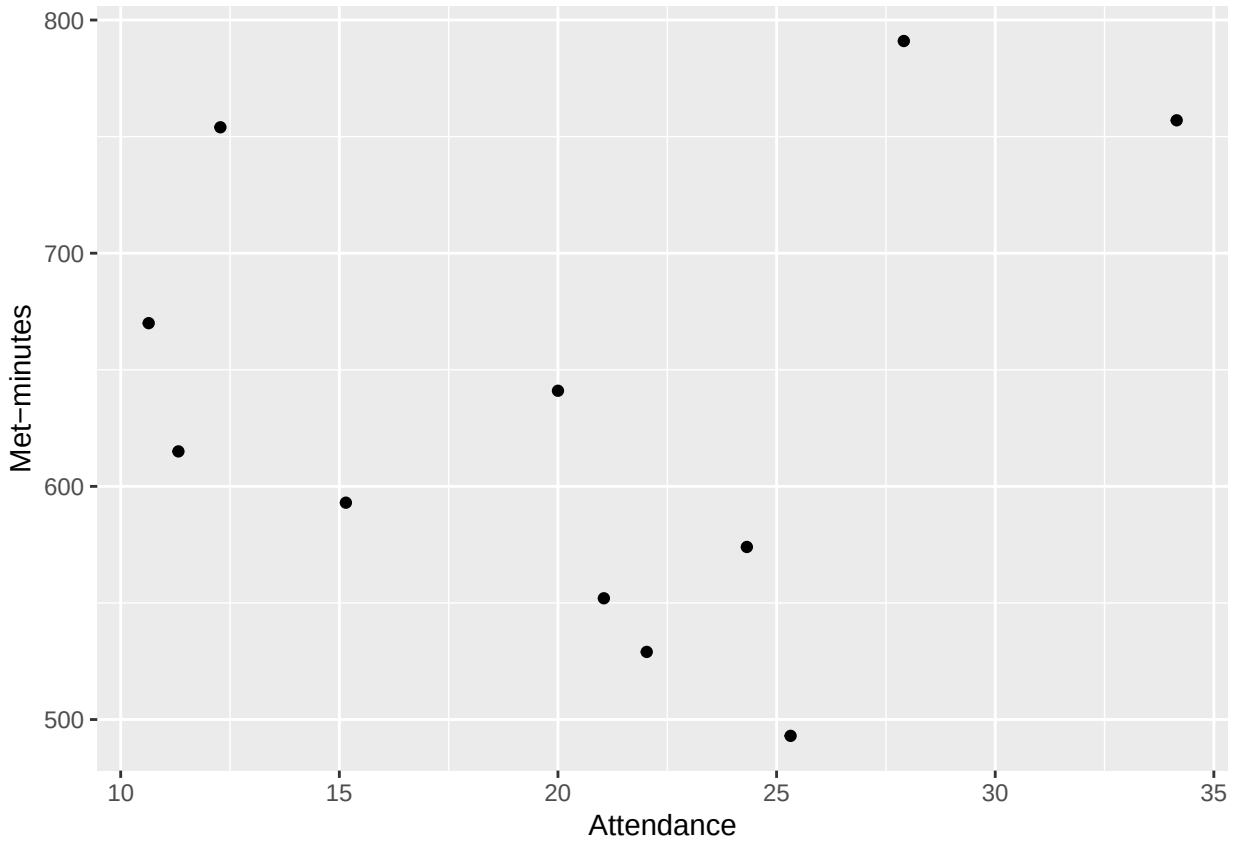
```
lmtest::lrtest(fullLm, half)
#> Likelihood ratio test
#>
#> Model 1: y ~ weeks + weeks * int
#> Model 2: y ~ weeks
#>   #Df  LogLik Df  Chisq Pr(>Chisq)
#> 1   5 -29.836
#> 2   3 -36.875 -2 14.078  0.0008771 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this case, the larger model which also regresses on the presence of the intervention, is statistically significantly better for our data. So the intervention had a significicant effect on program attendance. Wohoo.

Met-minute and % Attendance and interaction (because it is a sign of compliance)

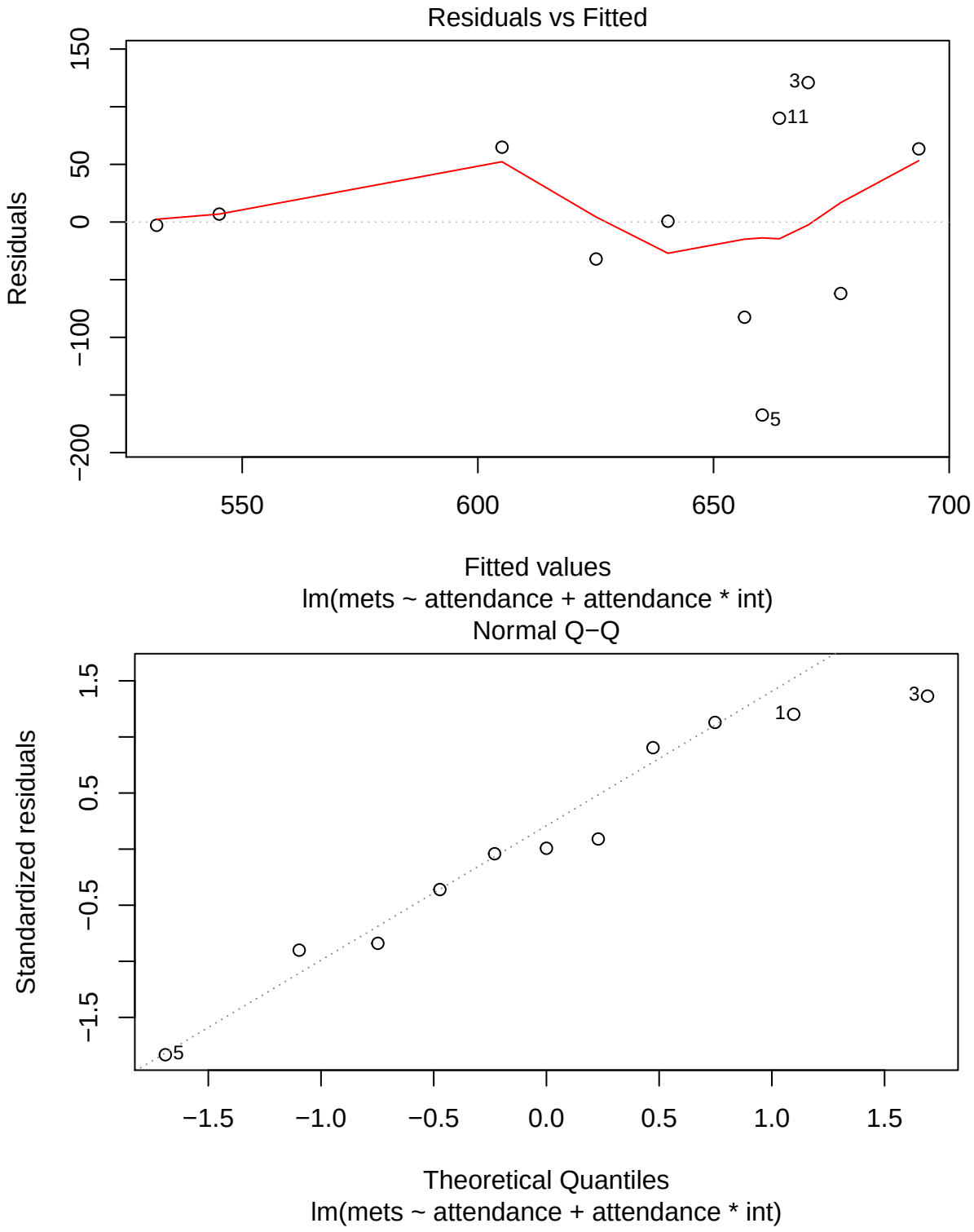
We want to see whether the intervention had a significant impact on the relationship between attendance and met-minute interactions. First, let's just see this relationship mapped out in the 2020 data.

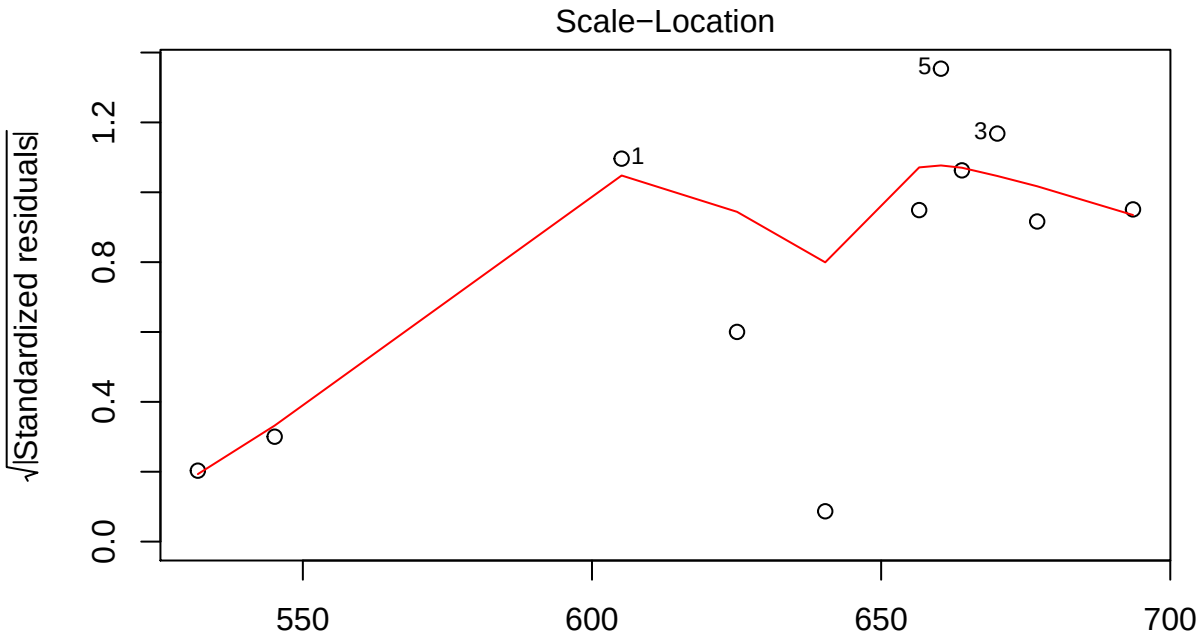
```
library(ggplot2)
#> Warning: package 'ggplot2' was built under R version 3.5.2
#Create the data encoded with a dummy variable for the pre/post occurence intervention.
ourDf <- data.frame(attendance = as.numeric(RMHF::read_group_data()[12:22,]$"% of patients who were no-s
ggplot(data = ourDf, mapping = aes(attendance, mets)) + geom_point() + labs(x = "Attendance", y = "Met-m
```

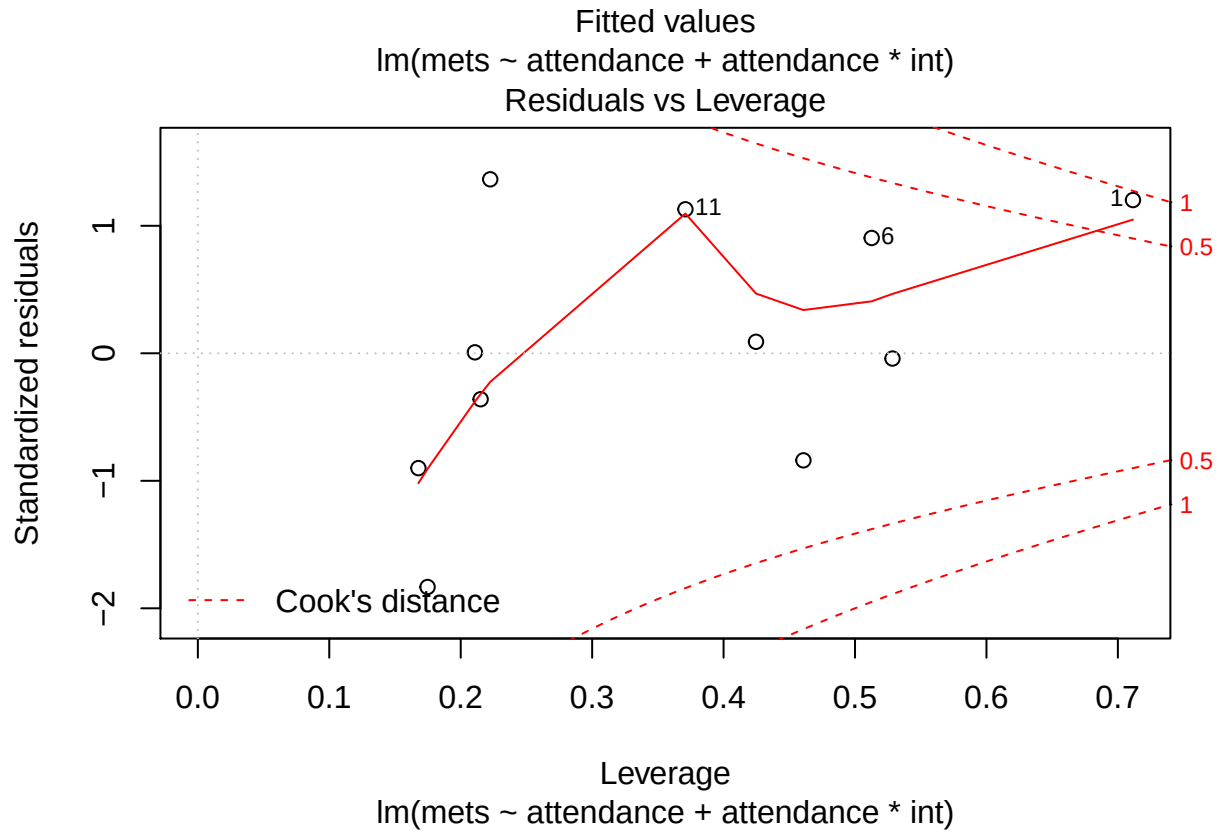


Now let's fit a regression line and check our assumptions.

```
fit <- lm(data = ourDf, formula = mets~ attendance + attendance*int)
print(fit)
#>
#> Call:
#> lm(formula = mets ~ attendance + attendance * int, data = ourDf)
#>
#> Coefficients:
#>    (Intercept)      attendance             int  attendance:int
#>        565.099           3.761         265.285         -17.313
plot(fit)
```


Residuals vs Fitted
Residuals
150
50
0
−100
−200
3
11
5
550
600
650
700
Fitted values
lm(mets ~ attendance + attendance * int)
Normal Q−Q
Standardized residuals
1.5
0.5
−0.5
−1.5
3
1
5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Theoretical Quantiles
lm(mets ~ attendance + attendance * int)






```
summary(fit)
#>
#> Call:
#> lm(formula = mets ~ attendance + attendance * int, data = ourDf)
#>
#> Residuals:
```

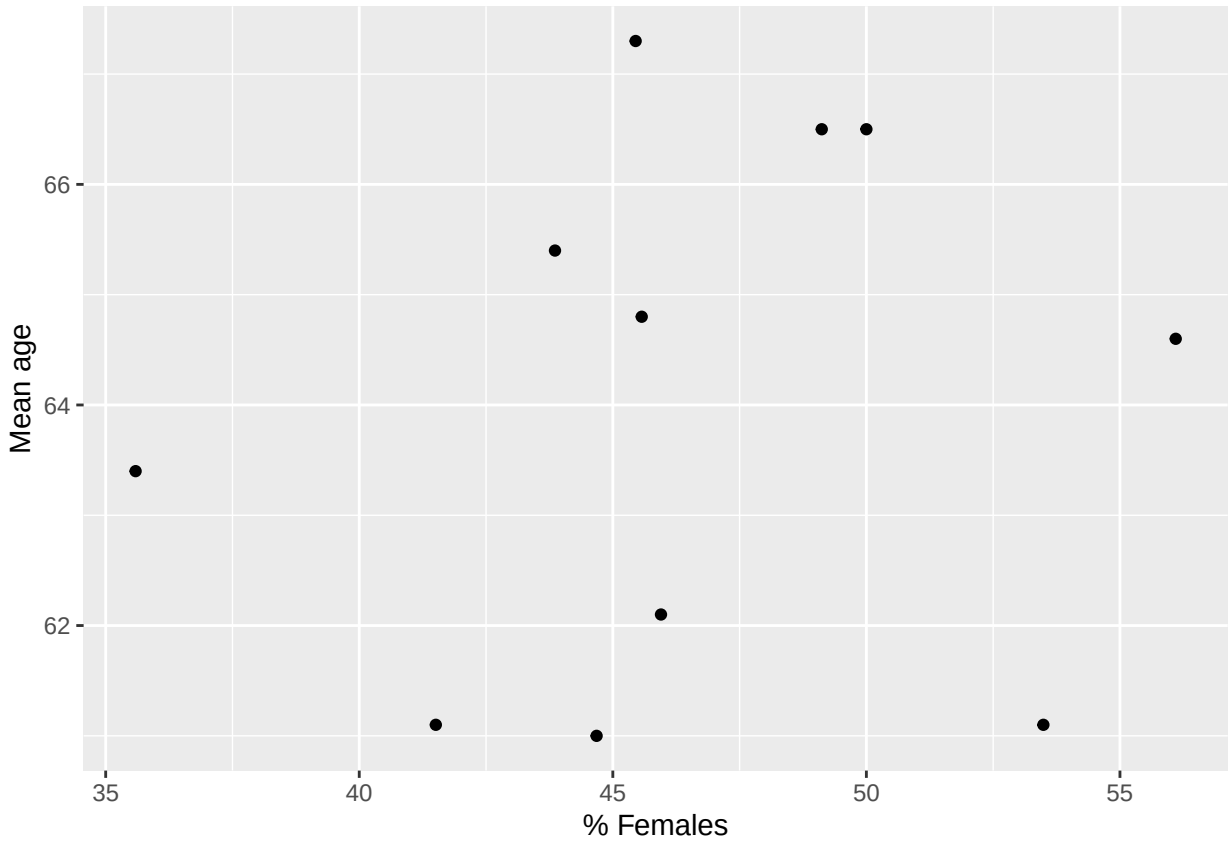
```
#>      Min       1Q   Median       3Q      Max
#> -167.339  -47.030    0.672   64.163  120.919
#>
#> Coefficients:
#>                Estimate Std. Error t value Pr(>|t|)
#> (Intercept)     565.099    140.634   4.018  0.00507 **
#> attendance        3.761      5.670   0.663  0.52832
#> int             265.285    222.493   1.192  0.27198
#> attendance:int  -17.313     11.644  -1.487  0.18065
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 100.5 on 7 degrees of freedom
#> Multiple R-squared:  0.2843, Adjusted R-squared:  -0.02248
#> F-statistic: 0.9267 on 3 and 7 DF,  p-value: 0.4763
```

We note the lack of statistical significance in the attendance*intervention coefficient, telling us that the effect of the intervention was not statistically significant.

Mean age and % females

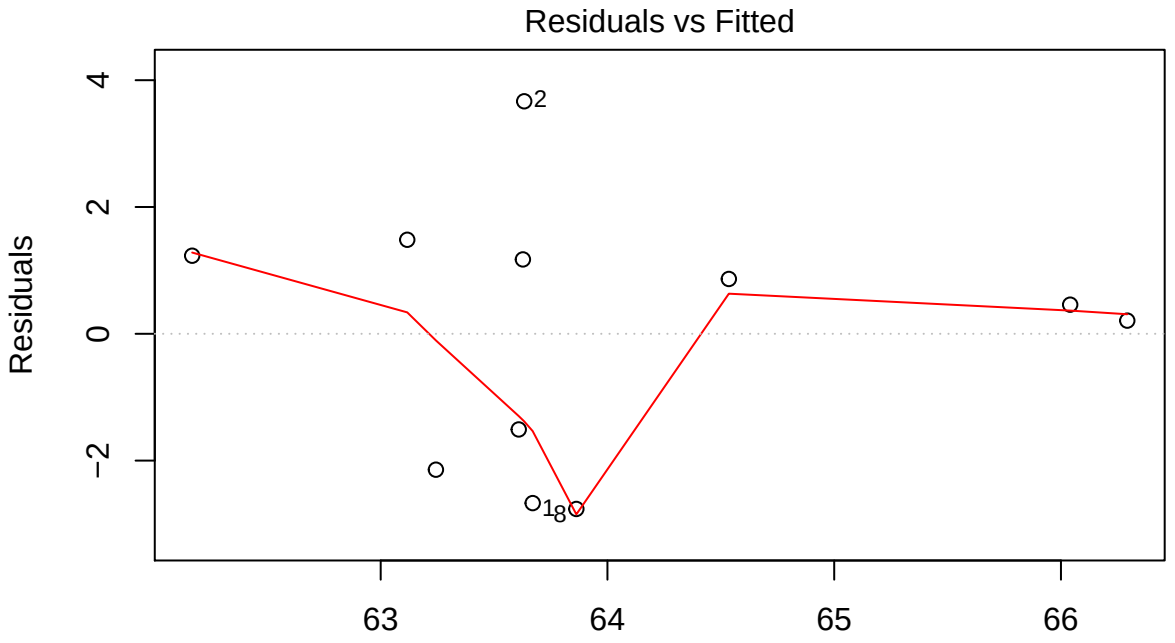
Now we'd like to see whether the intervention had a significant impact on the relationship between mean age and % females. First, let's just see this relationship mapped out in the 2020 data.

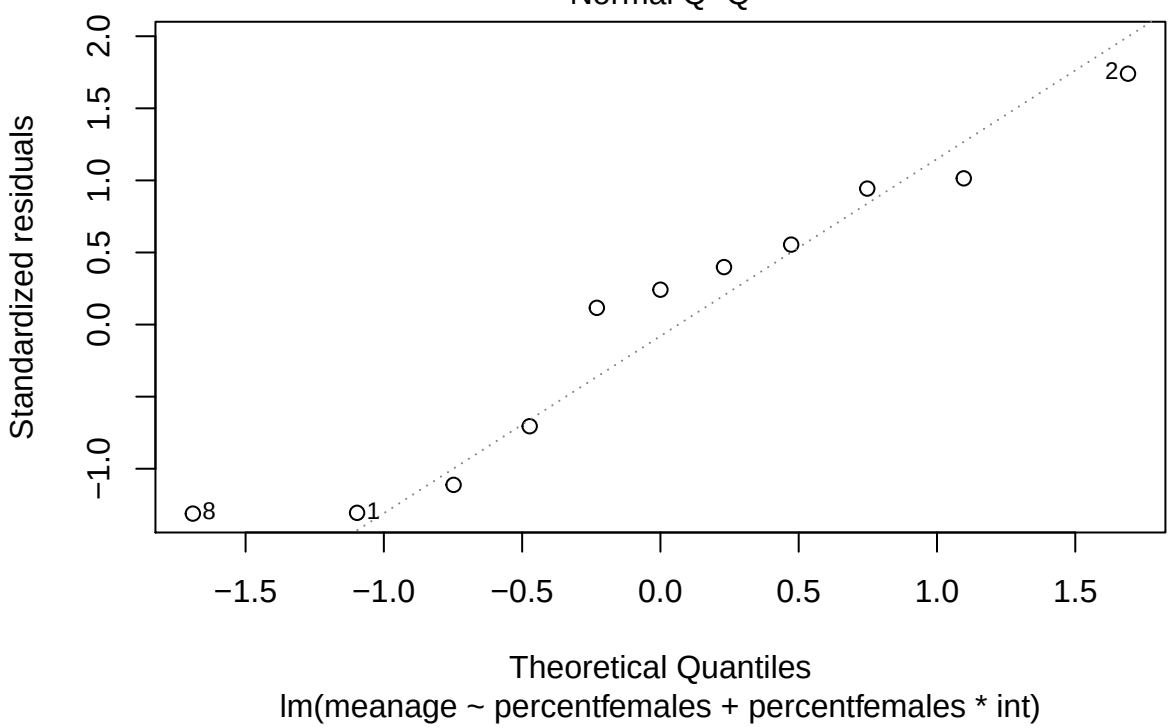
```
library(ggplot2)
#Create the data encoded with a dummy variable for the pre/post occurence intervention.
ourDf <- data.frame(percentfemales = as.numeric(RMHF::read_group_data()[12:22,]$"% Females"), meanage =
ggplot(data = ourDf, mapping = aes(percentfemales, meanage)) + geom_point() + labs(x = "% Females", y =
```

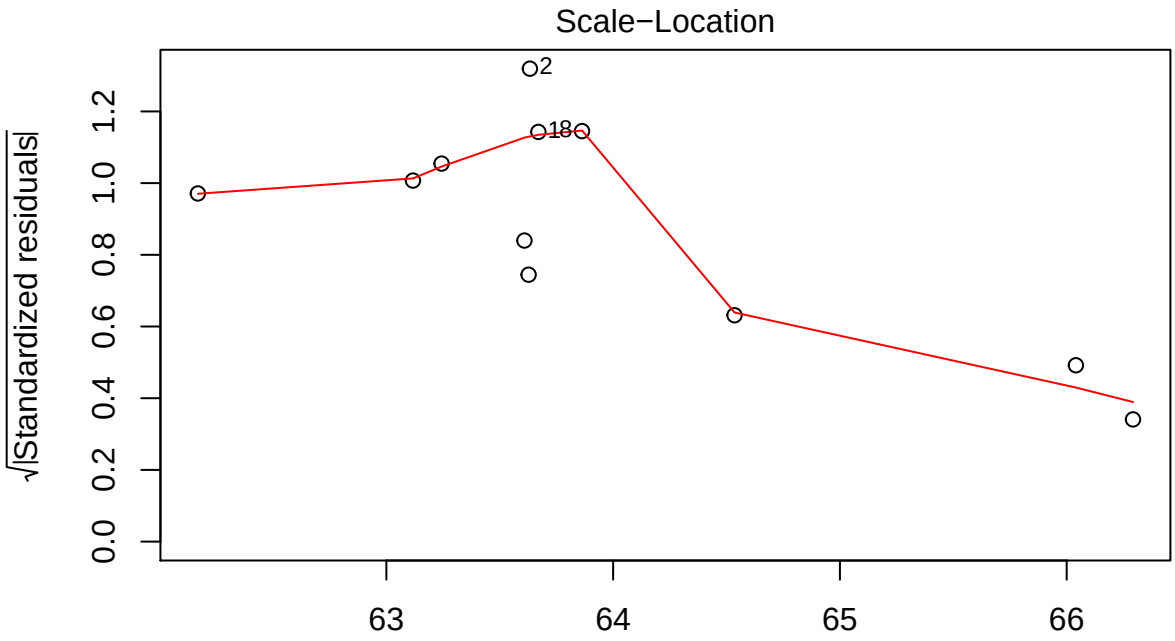


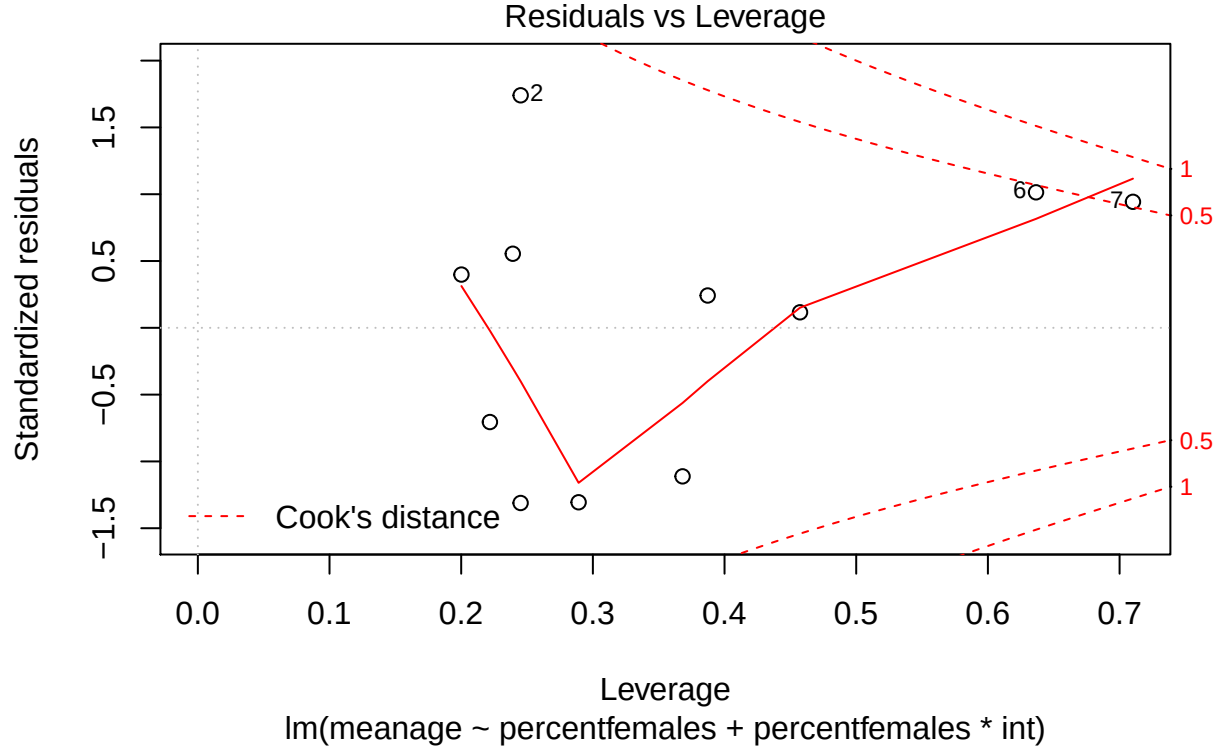
Now let's fit a regression line and check our assumptions.

```
fit <- lm(data = ourDf, formula = meanage ~ percentfemales + percentfemales*int)
print(fit)
#>
#> Call:
#> lm(formula = meanage ~ percentfemales + percentfemales * int,
#>     data = ourDf)
#>
#> Coefficients:
#>        (Intercept)      percentfemales                 int
#>           65.83425            -0.04843          -13.85060
#> percentfemales:int
#>            0.33461
plot(fit)
```


Residuals vs Fitted
Residuals
4
2
0
−2
2
18
63
64
65
66
Fitted values
lm(meanage ~ percentfemales + percentfemales * int)


Normal Q−Q
Standardized residuals
2.0
1.5
1.0
0.5
0.0
−1.0
2
8
1
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Theoretical Quantiles
lm(meanage ~ percentfemales + percentfemales * int)






```
summary(fit)
#>
#> Call:
#> lm(formula = meanage ~ percentfemales + percentfemales * int,
#>     data = ourDf)
#>
```

```
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -2.7628 -1.8262  0.4594  1.2021  3.6670
#>
#> Coefficients:
#>                     Estimate Std. Error t value Pr(>|t|)
#> (Intercept)         65.83425   10.72008   6.141 0.000472 ***
#> percentfemales      -0.04843    0.21991  -0.220 0.831968
#> int                -13.85060   14.07077  -0.984 0.357734
#> percentfemales:int   0.33461    0.30104   1.112 0.303065
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 2.425 on 7 degrees of freedom
#> Multiple R-squared:  0.2665, Adjusted R-squared:  -0.04781
#> F-statistic: 0.8479 on 3 and 7 DF,  p-value: 0.5102
```

We once again note the lack of statistical significance.

Conclusions

The lack of statistically signifiant decreases in almost every variable we looked at is telling. Despite so much changing during the COVID-19 Pandemic, much stayed constant. For instance, despite gyms shuttering their doors, the exercise before to the exercise after the pandemic was constant. This is interesting. Even more interesting is that despite the fact that appointments during the pandemic are fully virtual, attendance went down significantly during the pandemic. One would generally expect the opposite result, as virtual appointments do not require one to leave one's house.