Occupancy Forecaster

3/15/2022

Loading packages

Set Global Variables

```
#Global Variables
school = 'UVA'
room = '211 Olsson'
start_date = '2021-09-01 00:00:00'
end_date = '2021-12-01 00:00:00'
```

Load and Clean Motion Data

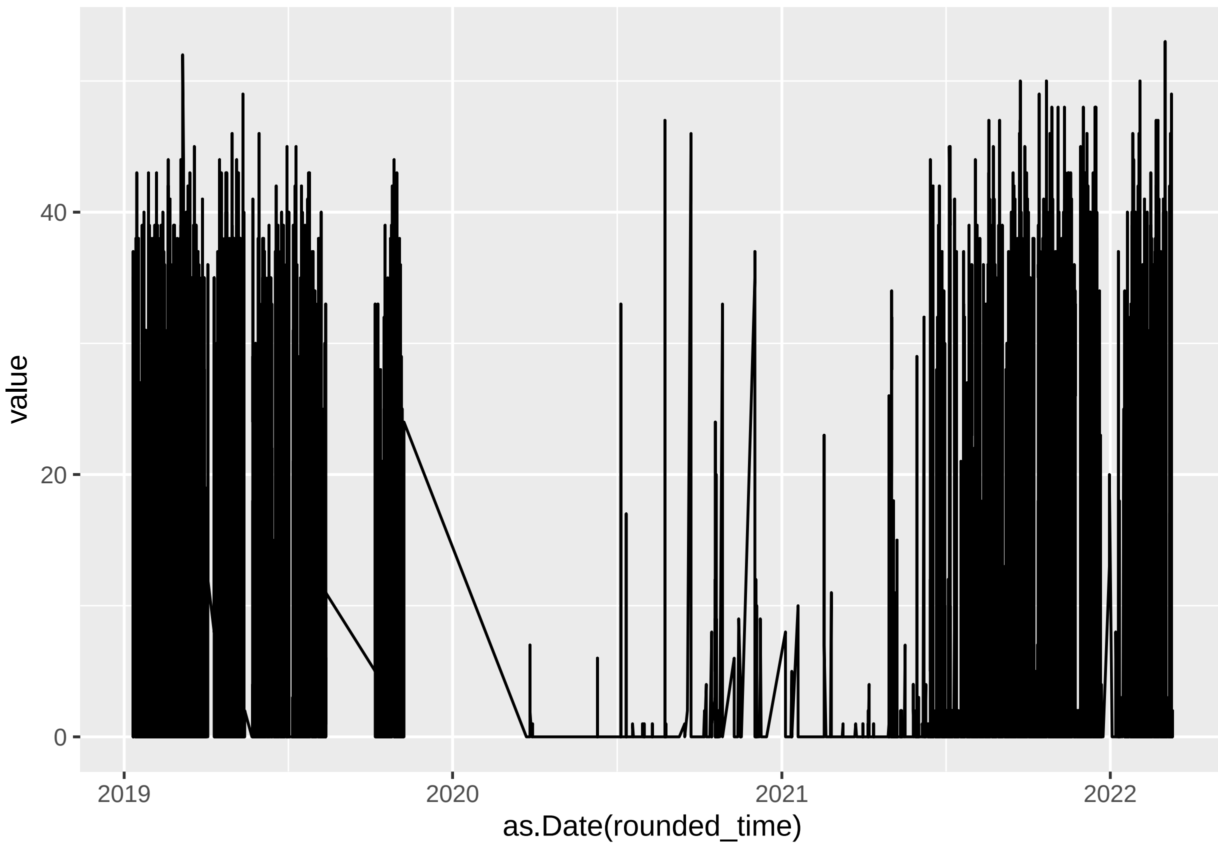
```
#Load Motion Data
motion_data = read.table('../Data/motion_data.csv',header=T,sep=',')

#Select Date and Value
motion_data = motion_data %>%
  filter(location_general==school && location_specific==room) %>%
  select(time,value)

#Remove Duplicate Time Stamps
motion_data = motion_data %>%  distinct(time, .keep_all = T)

#Convert Time String to Time Object and Strip Extra Content
motion_data$time = strptime(motion_data$time,format = '%Y-%m-%d %H:%M:%OS')
```

Bin Data into Hours and Sum

```
#Bin the Sum of Motion Values into same hour
grouped_motion_data = motion_data %>%
  mutate(rounded_time = trunc(time, "hours")) %>%
  group_by(rounded_time) %>%
  summarise(value=sum(value))
```

Plot all Data

```
#Plot the Time Series
grouped_motion_data %>%
  ggplot(aes(x=as.Date(rounded_time), y=value))+
  geom_line()
```

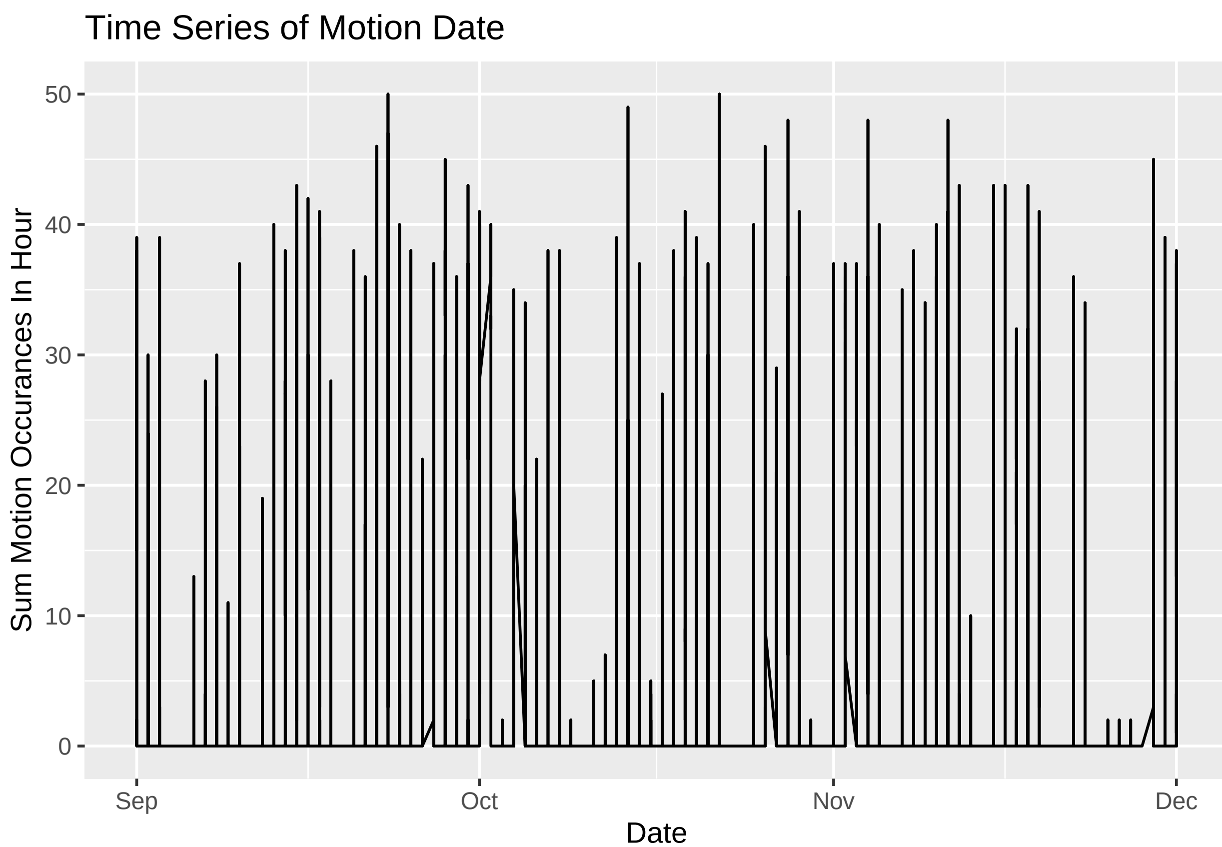


Select Starting Date

```
#Convert to Date
grouped_motion_data$rounded_time = as.Date(grouped_motion_data$rounded_time)

#Choose Start Date
grouped_motion_data = grouped_motion_data %>%
  filter(rounded_time >= as.Date(start_date) & rounded_time <= as.Date(end_date))
```

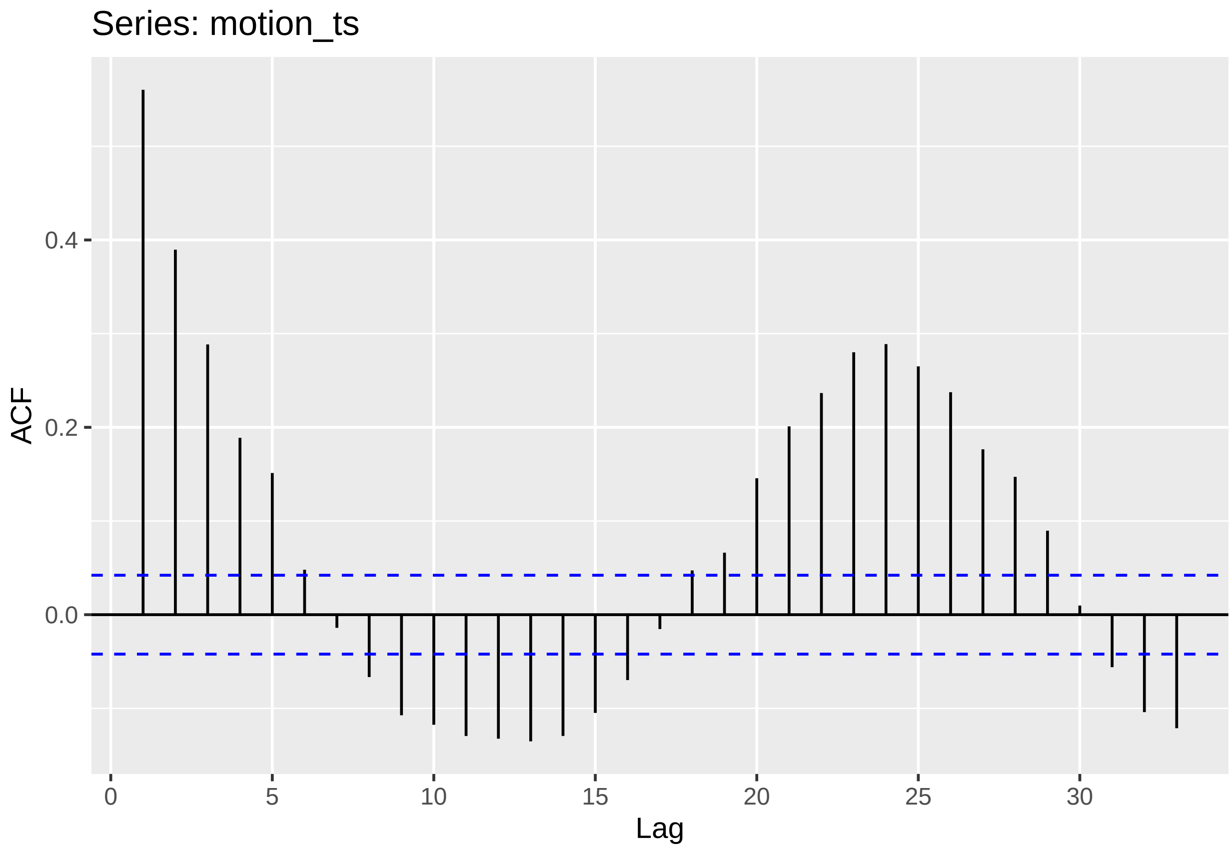
Plot Date Starting at New Date

```r
#Plot the Time Series
grouped_motion_data %>%
  ggplot(aes(x=as.Date(rounded_time), y=value))+
  geom_line()+
  ylab("Sum Motion Occurances In Hour")+
  xlab("Date")+
  labs(title="Time Series of Motion Date")
```



Investigate Seasonality with ACF

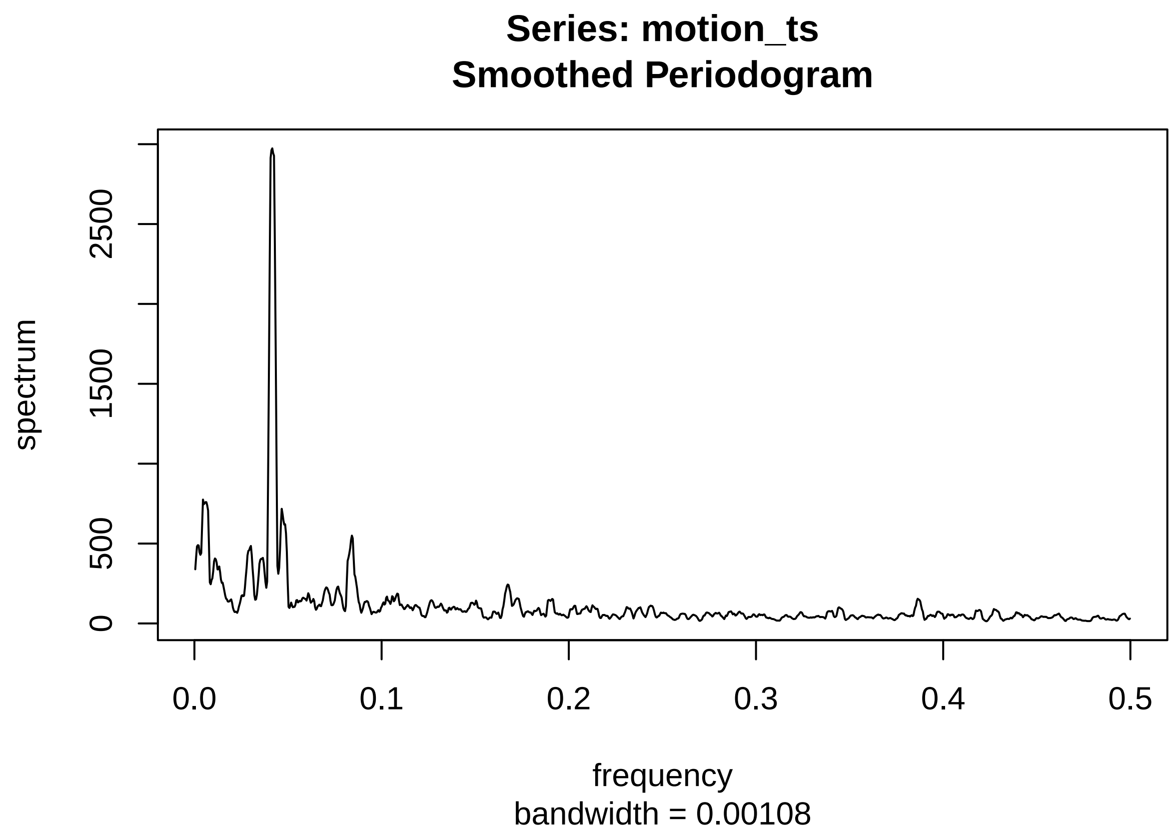
```r
#Make time series object
motion_ts = ts(grouped_motion_data$value)

#ACF
ggAcf(motion_ts)
```



Control for seasonality before controlling for trends. The Autocorrelation Function (ACF) shows the correlation of data points at different lags (correlation of points to their previous selves). We see repeated peaks. The dotted line shows that a previous point is significant in predicting the future. We need to remove seasonality by modeling it.

Investigate Seasonality using a Periodogram

```
motion_periodogram = spec.pgram(motion_ts, spans=9, demean = T, log='no')
```




```r
#Find the max peak
max_omega =
  motion_periodogram$freq[which(motion_periodogram$spec==max(motion_periodogram$spec))]

#The peak is:
max_omega
```

```
## [1] 0.04160951
```

```r
#The period is:
1/max_omega
```

```
## [1] 24.03297
```

Seasons repeating every 24 hours (1 day). This is a daily season. Since we observe smoothly varying seasons, we can capture seasonality by using trigonometric functions.

Build A Model to Capture Seasonality

```r
#Make T
t = c(seq(1,dim(grouped_motion_data)[1]))
#FIX ME (Must change if you remove points for forecasting)
```

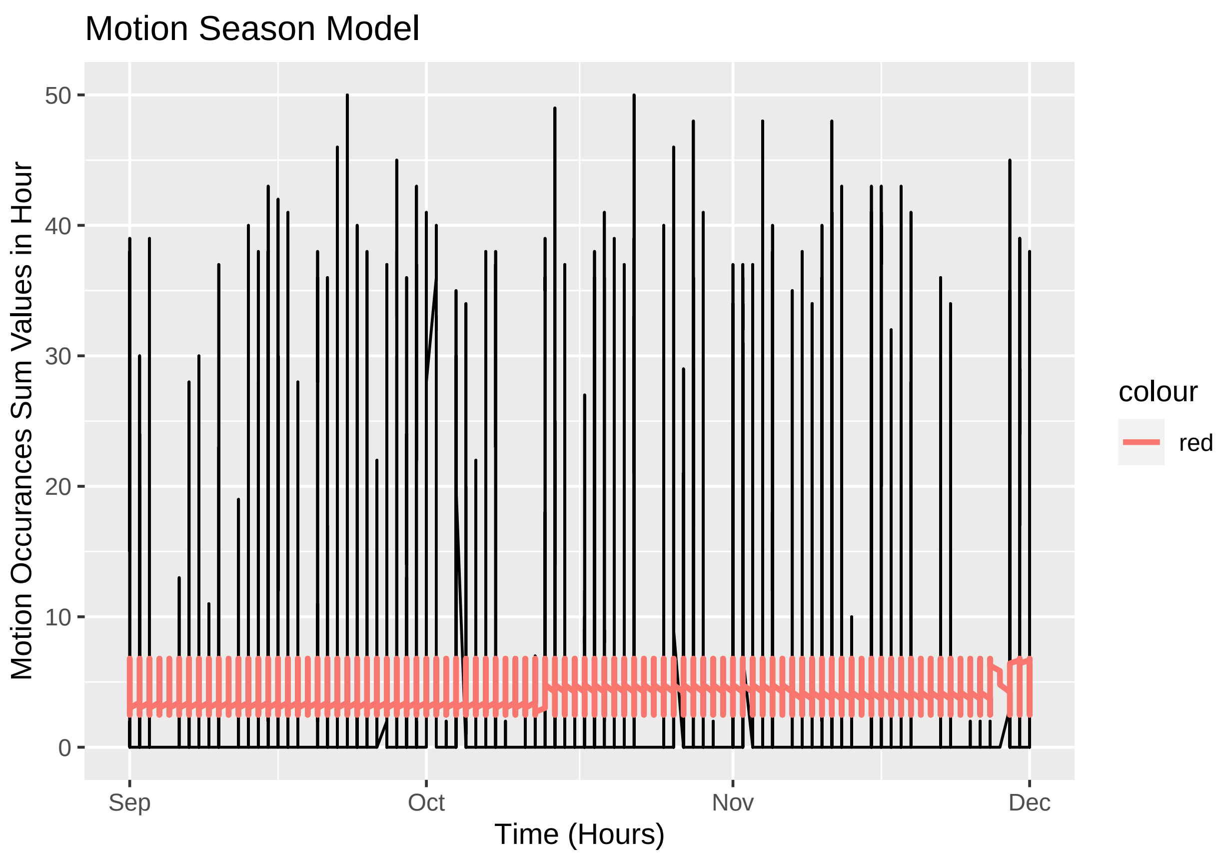
```r
motion_season_model = lm(motion_ts ~ sin(2*pi*t/24) + cos(2*pi*t/24))
summary(motion_season_model)
```

```
##
## Call:
## lm(formula = motion_ts ~ sin(2 * pi * t/24) + cos(2 * pi * t/24))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.793 -5.583 -3.429 -2.518 46.571
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)          4.6384     0.2299  20.177  < 2e-16 ***
## sin(2 * pi * t/24)   1.4152     0.3250   4.355 1.39e-05 ***
## cos(2 * pi * t/24)  -1.6318     0.3252  -5.017 5.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.7 on 2164 degrees of freedom
## Multiple R-squared:  0.01996,    Adjusted R-squared:  0.01905
## F-statistic: 22.04 on 2 and 2164 DF,  p-value: 3.36e-10
```

Accounting for seasonality is significant

Visualize Model Capturing Seasonality

```r
grouped_motion_data %>%
  ggplot(aes(x=rounded_time, y=value))+
  geom_line()+
  geom_line(aes(x=rounded_time,
              y=motion_season_model$fitted.values, color="red"), size=1)+
  ylab("Motion Occurances Sum Values in Hour")+
  xlab("Time (Hours)")+
  labs(title="Motion Season Model")
```



Investigate Trends

```
motion_trend_model = lm(motion_ts ~ t)
summary(motion_trend_model)
```

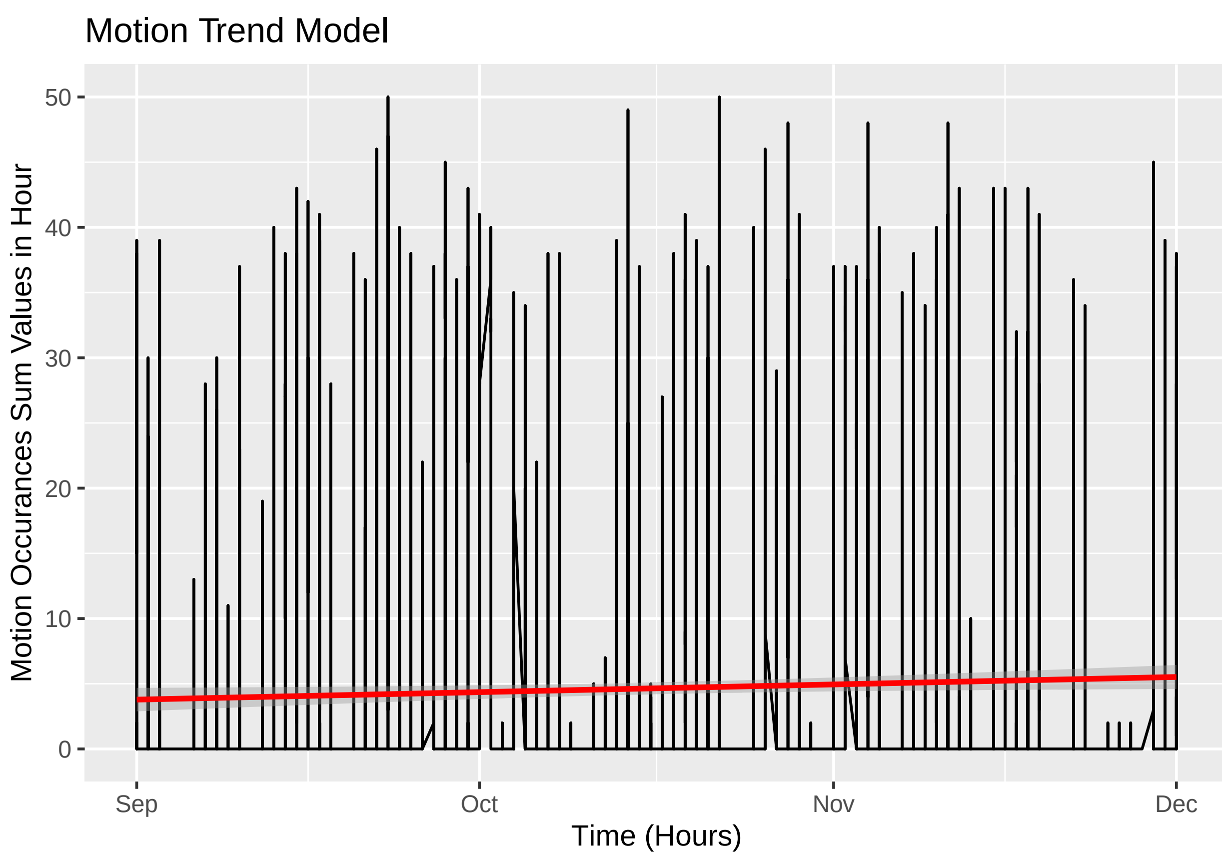
```
##
## Call:
## lm(formula = motion_ts ~ t)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.521 -4.930 -4.343 -3.793 45.800
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.7569843  0.4639612   8.098 9.25e-16 ***
## t           0.0008142  0.0003707   2.196   0.0282 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.8 on 2165 degrees of freedom
## Multiple R-squared:  0.002223,   Adjusted R-squared:  0.001762
## F-statistic: 4.824 on 1 and 2165 DF,  p-value: 0.02817
```

Time is a significant predictor. Therefore, we can capture trends by incorporating timesteps into our model

Plot Model that Captures Trends

```
grouped_motion_data %>%
  ggplot(aes(x=rounded_time, y=value))+
  geom_line()+
  stat_smooth(method='lm',col="red")+
  ylab("Motion Occurances Sum Values in Hour")+
  xlab("Time (Hours)")+
  labs(title="Motion Trend Model")
```

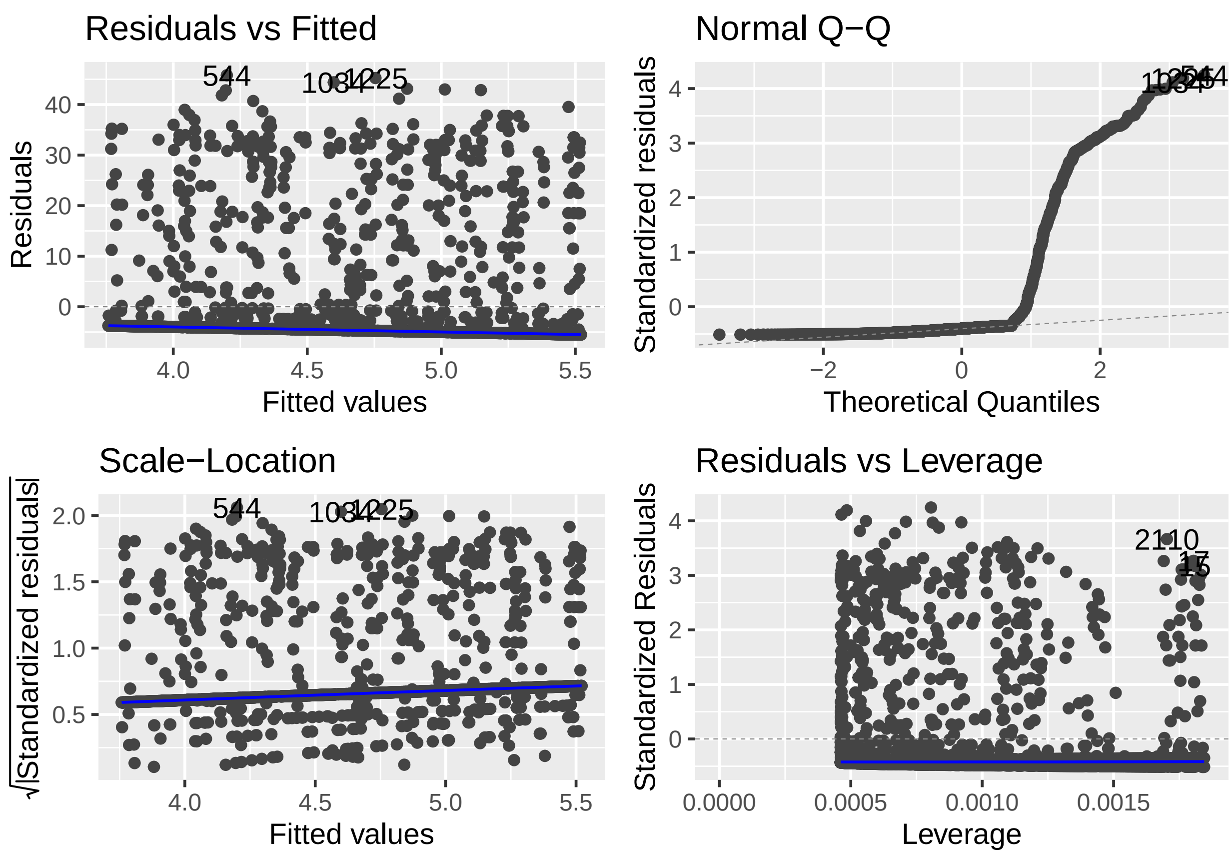
```
## `geom_smooth()` using formula 'y ~ x'
```



Slight increase likely due to more students on grounds after COVID.

Trend Model Diagnostics Sanity Check

```
autoplot(motion_trend_model, labels.id=NULL)
```



Non-guassian tail.

Capture Seasonality and Trends

```
motion_season_trend_model = lm(motion_ts ~ t + sin(2*pi*t/24) + cos(2*pi*t/24))
summary(motion_season_trend_model)
```

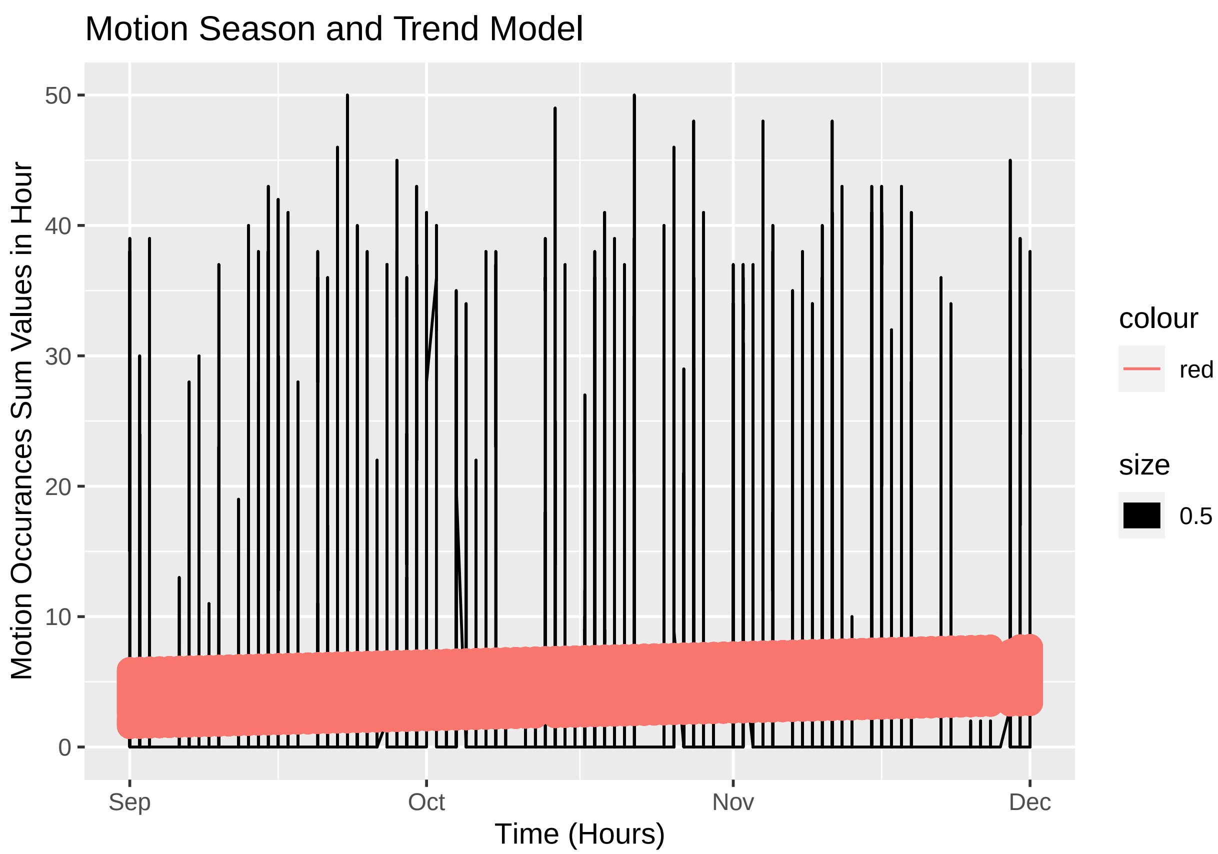
```
##
## Call:
## lm(formula = motion_ts ~ t + sin(2 * pi * t/24) + cos(2 * pi *
##     t/24))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.674 -5.486 -3.562 -2.155 46.458
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         3.7421737  0.4594991   8.144 6.39e-16 ***
## t                   0.0008268  0.0003671   2.252   0.0244 *
## sin(2 * pi * t/24)  1.4170837  0.3246621   4.365 1.33e-05 ***
## cos(2 * pi * t/24) -1.6351129  0.3249428  -5.032 5.25e-07 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.69 on 2163 degrees of freedom
## Multiple R-squared:  0.02225,    Adjusted R-squared:  0.0209
## F-statistic: 16.41 on 3 and 2163 DF,  p-value: 1.523e-10
```

Time and Seasonality are significant predictors of motion

Plot Season and Trend Model

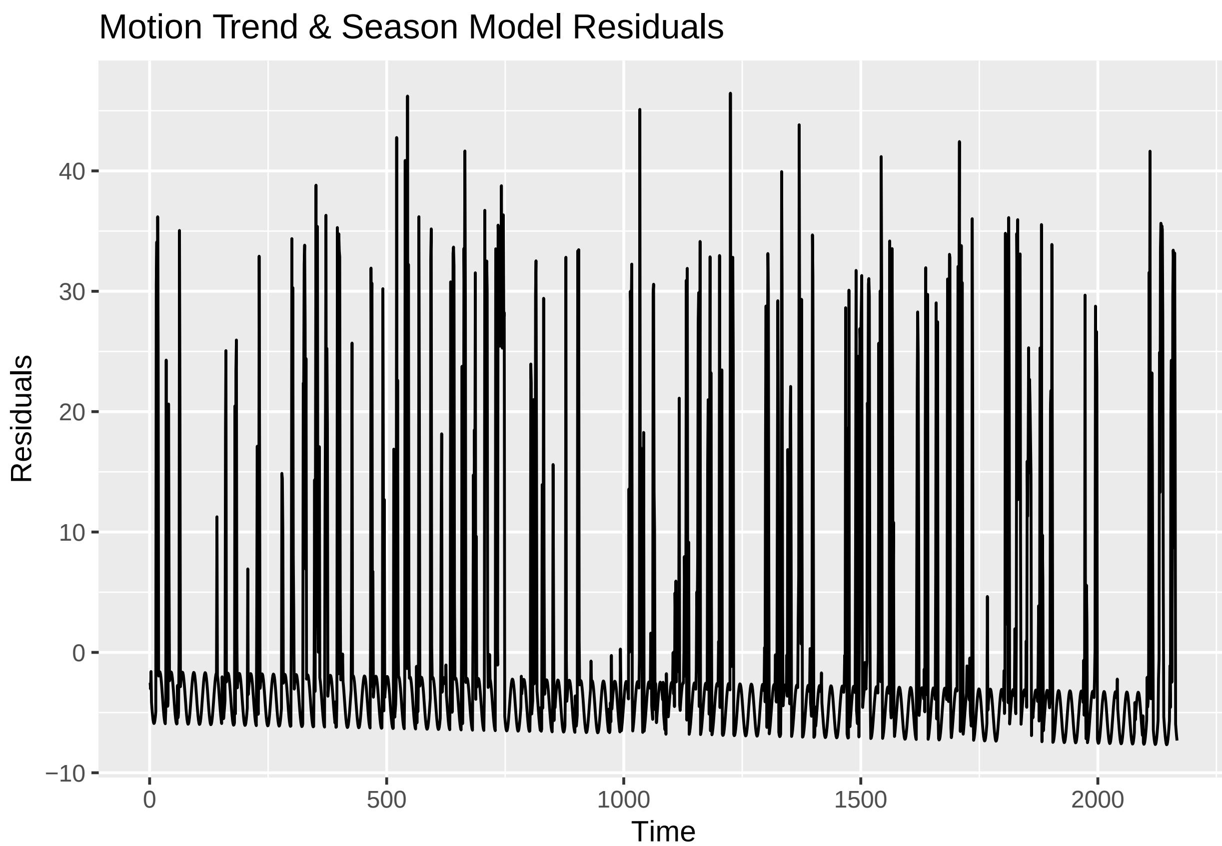
```
grouped_motion_data %>%
  ggplot(aes(x=rounded_time, y=value))+
  geom_line()+
  geom_line(aes(x=rounded_time,
                y=motion_season_trend_model$fitted.values,
                color="red",size=.5))+
  ylab("Motion Occurances Sum Values in Hour")+
  xlab("Time (Hours)")+
  labs(title="Motion Season and Trend Model")
```



Determine Auto-Regressive and Moving Average Components

Plot Residuals of Previous Model

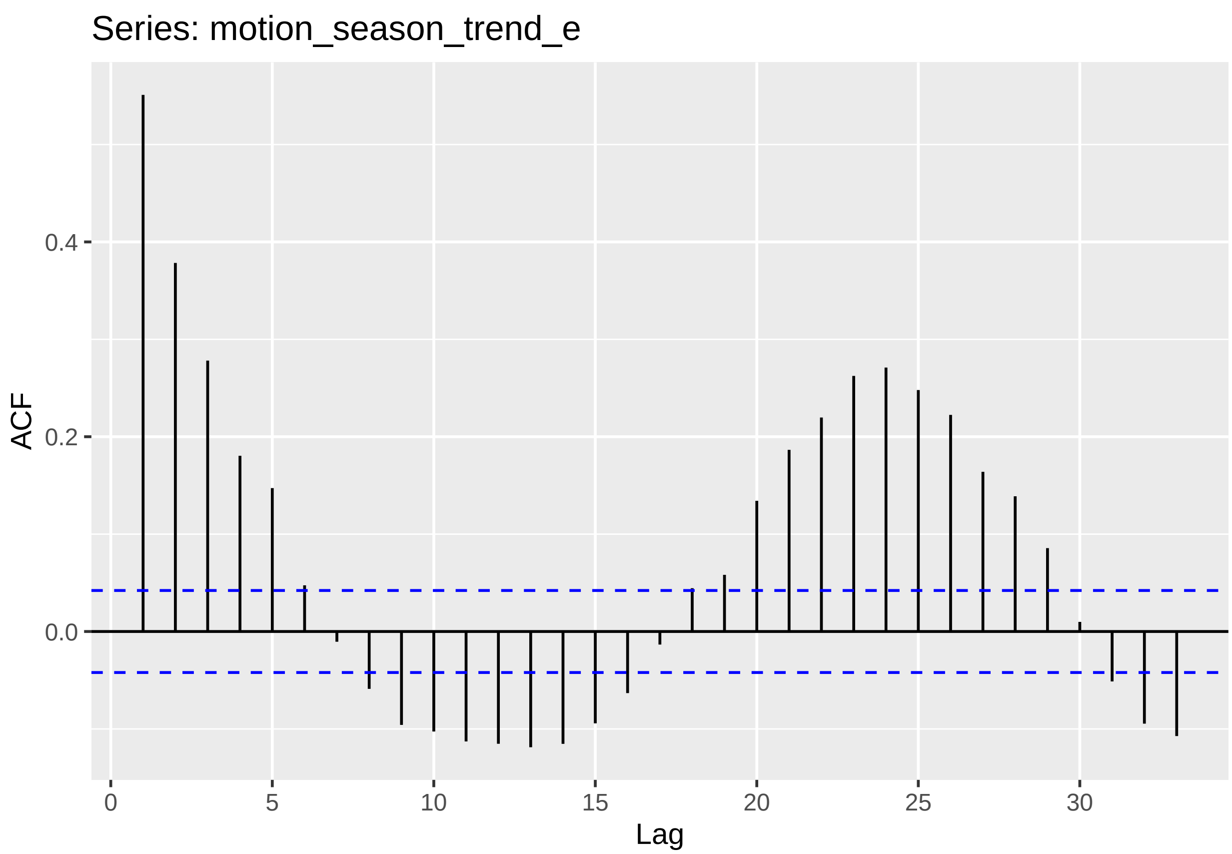
```
motion_season_trend_e = ts(motion_season_trend_model$residuals)
#plot residuals
autoplot(motion_season_trend_e, xlab="Time", ylab="Residuals")+
  labs(title="Motion Trend & Season Model Residuals")
```



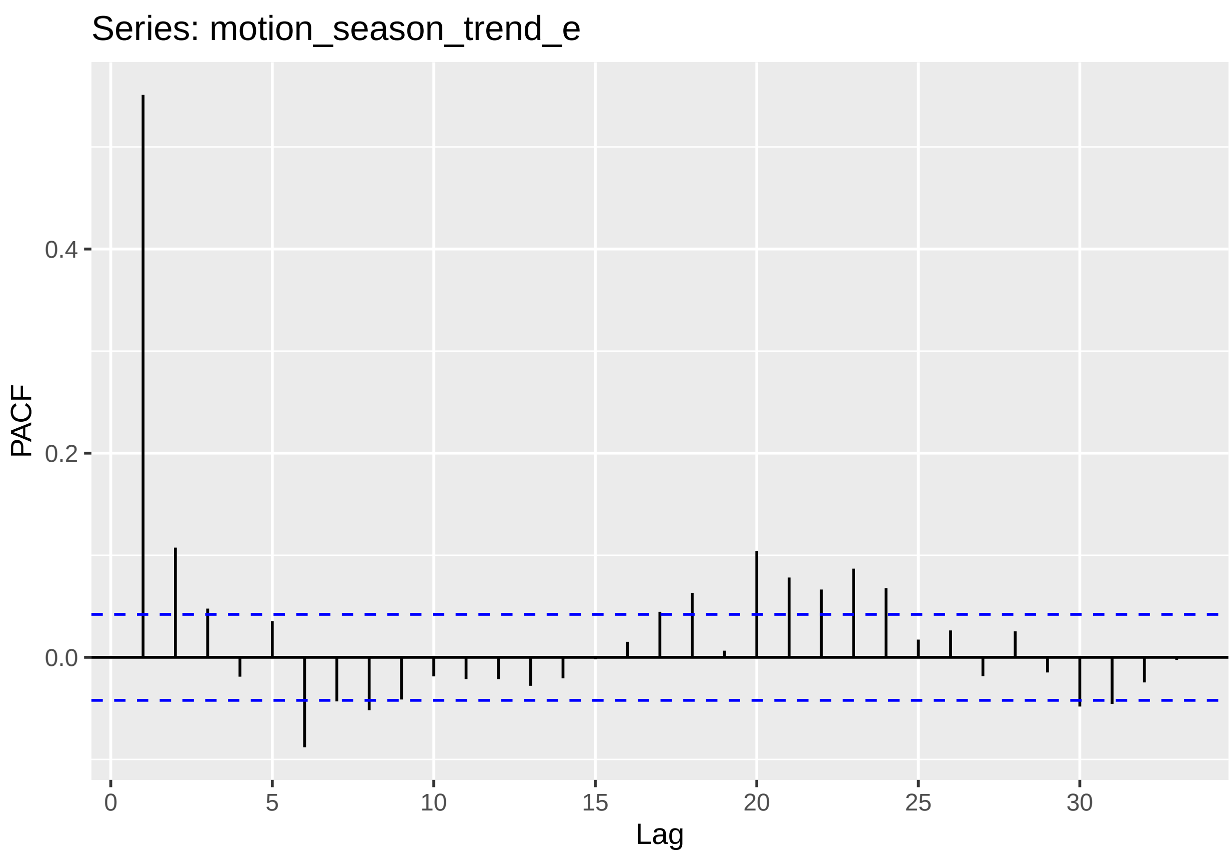
Modeling for trends and seasonality does not account for autocorrelation. To account for the residuals of our model that are not IID use and AR model. We must know how far back our residuals have "memory" by using an ACF and PACF.

ACF and PACF on the Residuals of Model

```
ggAcf(motion_season_trend_e)
```




```
ggPacf(motion_season_trend_e)
```



Since both the ACF and the PACF look sinusoidal, we should use an ARMA model. Since the PACF becomes insignificant after the 2nd lag, use AR(2). Since the ACF becomes insignificant after the 5th lag, use MA(5)

Build ARMA Model

```
motion_arma2_5 = arima(motion_season_trend_e, order =c(2,0,5))
summary(motion_arma2_5)
```

```
##
## Call:
## arima(x = motion_season_trend_e, order = c(2, 0, 5))
##
## Coefficients:
##          ar1      ar2      ma1     ma2     ma3     ma4     ma5  intercept
##       0.7411  -0.1967  -0.2532  0.1517  0.1159  0.0300  0.1199    -0.0242
## s.e.  0.3678   0.2721   0.3673  0.1063  0.0383  0.0292  0.0258     0.4831
##
## sigma^2 estimated as 77.59:  log likelihood = -7789.9,  aic = 15597.8
##
## Training set error measures:
##                        ME     RMSE     MAE       MPE     MAPE     MASE
## Training set 0.006267409 8.808725 4.98326 -14.22364 159.3939 1.117425
```

```
##                          ACF1
## Training set -0.0006382646
```

```r
AIC(motion_arma2_5) #15597.8
```

```
## [1] 15597.8
```

#Autoselect Model

```r
motion_auto = auto.arima(motion_season_trend_e)
summary(motion_auto)
```

```
## Series: motion_season_trend_e
## ARIMA(3,0,1) with zero mean
##
## Coefficients:
##           ar1     ar2     ar3     ma1
##       -0.2365  0.4381  0.1282  0.7265
## s.e.   0.1015  0.0566  0.0215  0.1006
##
## sigma^2 estimated as 78.25:  log likelihood=-7797
## AIC=15604   AICc=15604.03   BIC=15632.41
##
## Training set error measures:
##                        ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -0.002277777 8.837733 4.980406 -4.039215 148.7043 1.116785
##                       ACF1
## Training set 0.0008001257
```

```r
#ARIMA(3,0,1) 15604
```

The automatic selection chose ARMA(3,1) which performed worse than an ARMA(2,5) in terms of AIC.

```r
forecast1 = predict(motion_auto,30)
```