Project 2

Rachael Stryker, Tom Muhlbauer, and Daniel Lassiter

12/4/2020

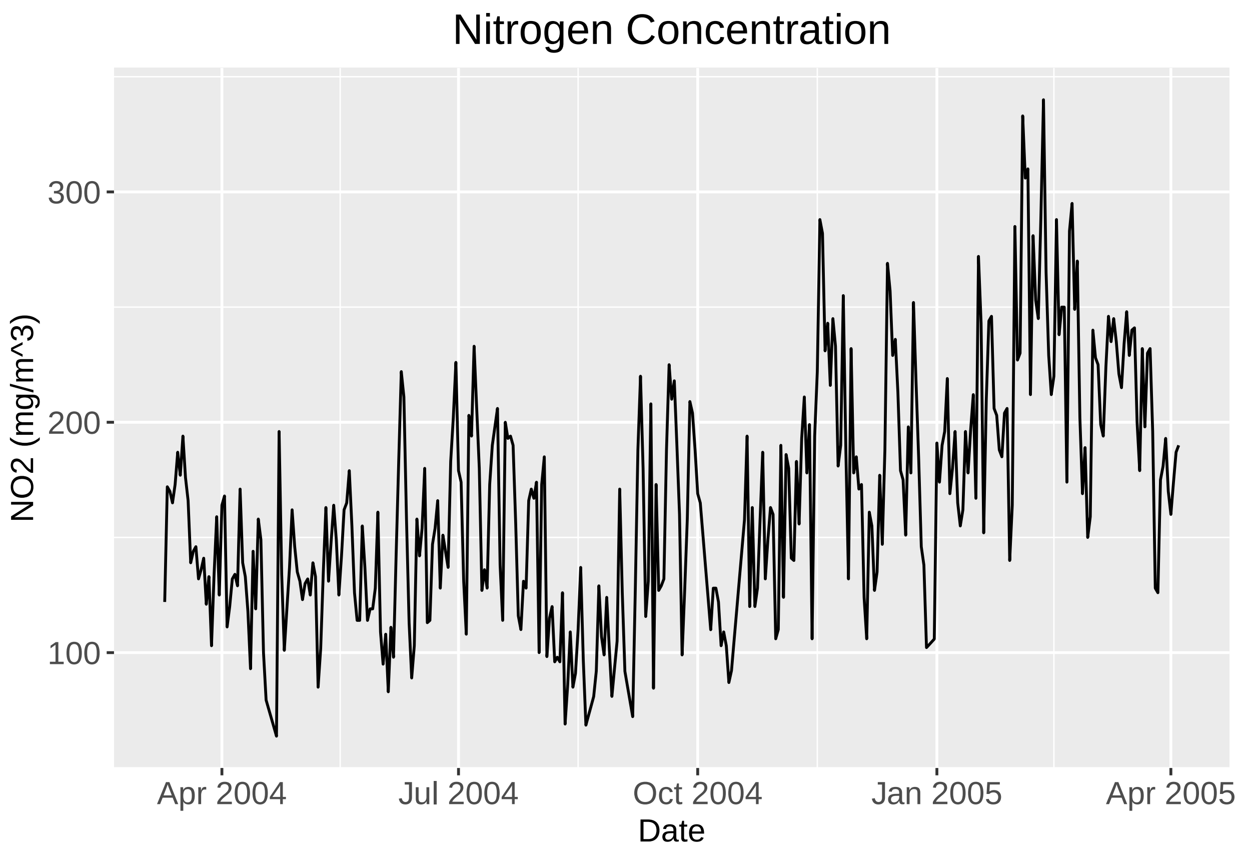
Contents

1 Exploratory Data Analysis

1.1 Inspecting for trends and seasonality

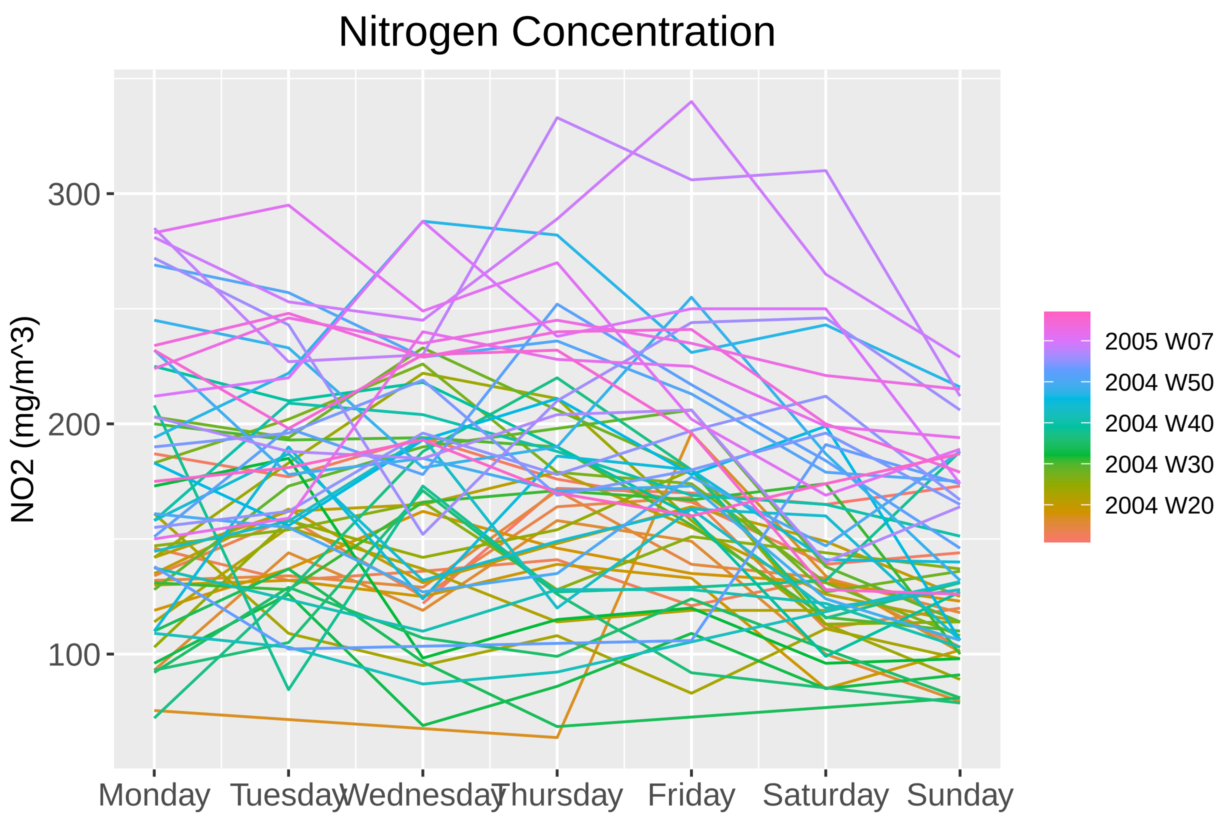
```
# For quickly formatting plots
my_theme <- theme(plot.title = element_text(hjust = 0.5, size = 16),
        axis.title = element_text(size = 12),
        axis.text = element_text(size = 12),
        plot.caption = element_text(size = 12, hjust = 0))
```

```
dailyAQ %>% autoplot(NO2) + ylab("NO2 (mg/m^3)") + xlab("Date") +
  ggtitle("Nitrogen Concentration") + my_theme
```



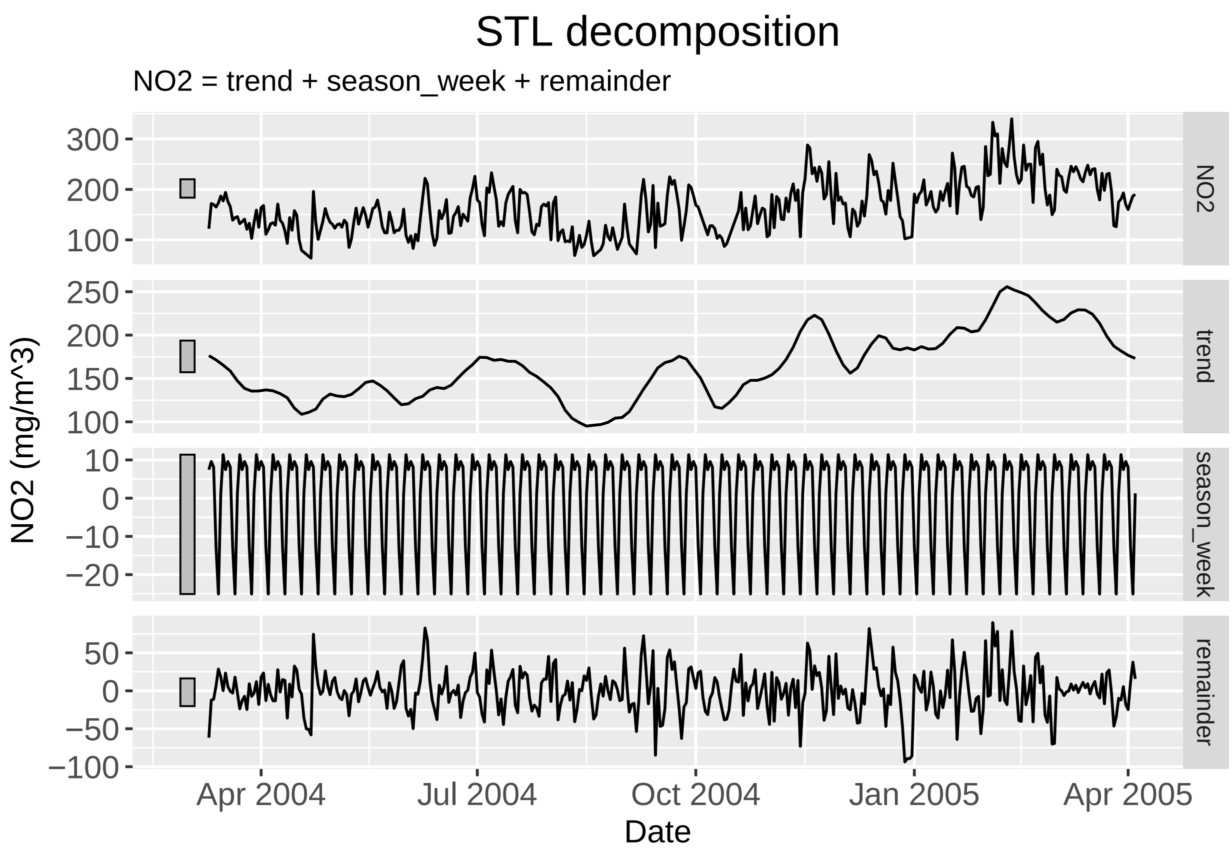
The time series plot indicates potential seasonal components and a potential trend component.

```
dailyAQ %>% gg_season(NO2, period = "week") + ylab("NO2 (mg/m^3)") + xlab("") +
    ggtitle("Nitrogen Concentration") + my_theme
```



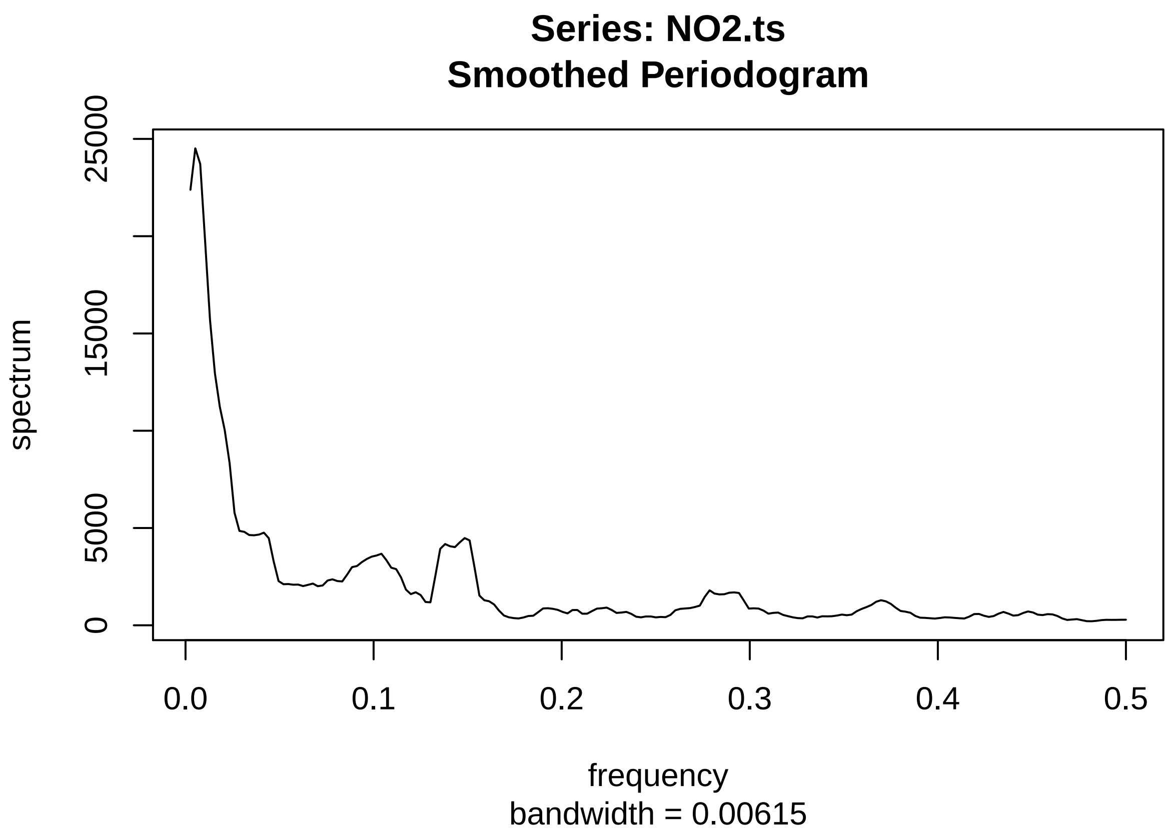
This plot indicates an increasing trend over time.

```
dailyAQ %>% model(STL(NO2 ~ trend(window=21) + season(window = "periodic"),
                     robust = TRUE)) %>%
                     components() %>%
                     autoplot() + my_theme + ylab("NO2 (mg/m^3)")
```



This seasonal decomposition chart shows a possibly increasing trend and a weekly seasonal component.

```r
NO2.ts <- head(NO2.ts, n = length(NO2.ts) - 8)

pg.NO2 <- spec.pgram(NO2.ts,spans=9,demean=T,log='no')
```




```
spec.NO2 <- data.frame(freq=pg.NO2$freq, spec=pg.NO2$spec)

spec.NO2 <- data.frame(freq=pg.NO2$freq, spec=pg.NO2$spec)%>%
  mutate(period = 1/freq) %>% arrange(desc(spec))

spec.NO2[c(1:17), ]
##           freq      spec     period
## 1  0.005208333 24514.627 192.000000
## 2  0.007812500 23711.028 128.000000
## 3  0.002604167 22385.338 384.000000
## 4  0.010416667 19749.539  96.000000
## 5  0.013020833 15692.398  76.800000
## 6  0.015625000 12945.511  64.000000
## 7  0.018229167 11232.528  54.857143
## 8  0.020833333 10035.151  48.000000
## 9  0.023437500  8340.280  42.666667
## 10 0.026041667  5785.995  38.400000
## 11 0.028645833  4849.327  34.909091
## 12 0.031250000  4803.541  32.000000
## 13 0.041666667  4760.502  24.000000
## 14 0.039062500  4662.202  25.600000
## 15 0.033854167  4638.409  29.538462
## 16 0.036458333  4627.030  27.428571
## 17 0.148437500  4482.699   6.736842
```

There is a peak at around .15 which corresponds to a seasonality component with a period of 1 week. There is also a possible monthly season and longer seasonal periods are also possible. We decided to model many combinations of these periods.

1.2 Hypotheses from exploratory data analysis:

There is an increasing trend in NO2 emissions. This could be due to increasing cars on the road over time. There is also one or more seasonal component.

2 Building Univariate Time Series Models

```
# Creating potential predictors
dailyAQ <- dailyAQ %>% mutate(step = 1:length(Date)) %>%
      mutate(day = as.factor(as.character(lubridate::wday(Date, label  = TRUE)))) %>%
      mutate(weekend = isWeekend(Date))

# Create train and test sets
dailyAQ_train <- dailyAQ %>% head(n = length(Date) - 8)
dailyAQ_test <- dailyAQ %>% tail(n = 7)
```

2.1 Modeling trend and season

Deciding how to model the seasonal component was an iterative process. The models shown below include only those we considered to be reasonable candidates. Note that we decided to use the packages feasts, fable, and fabletools for these analyses so our code looks different than the the in class examples.

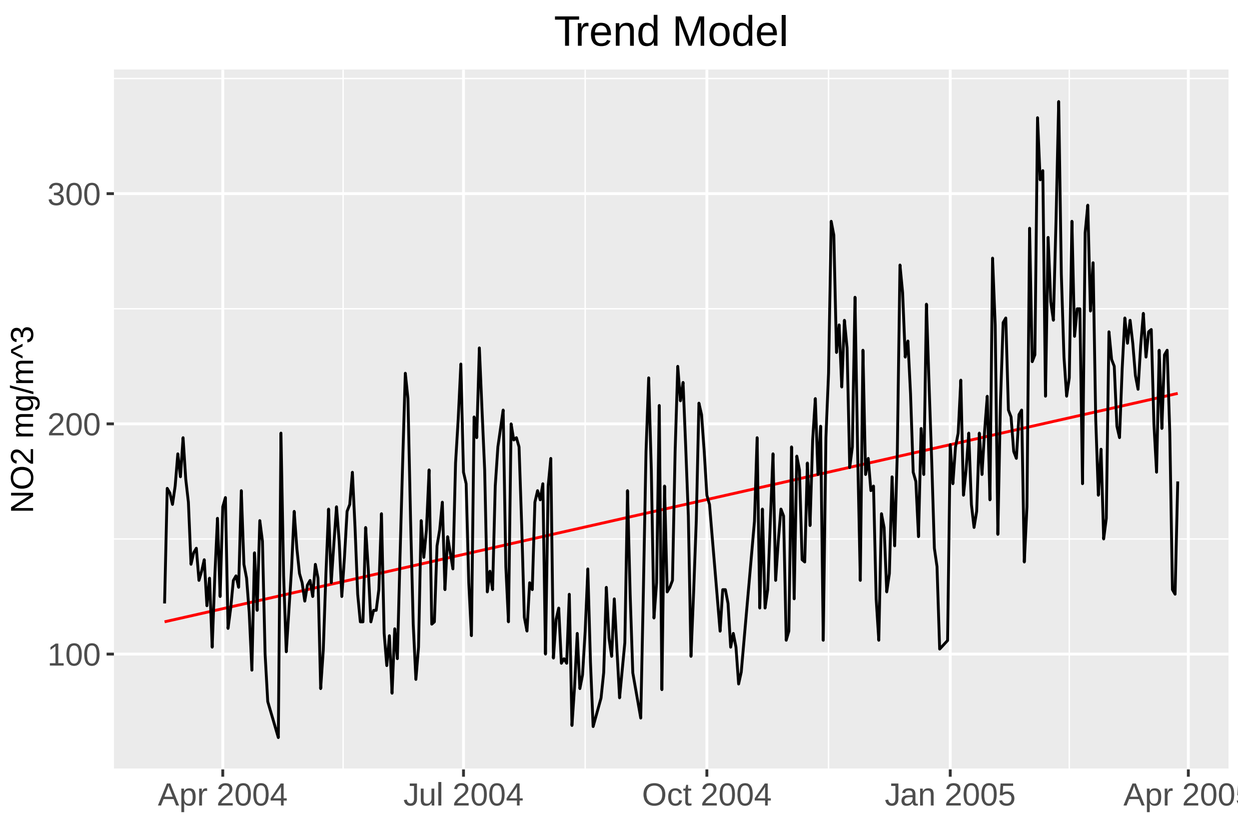
```
# Create potential trend/season models
models <- dailyAQ_train %>%
  model(trend = TSLM(NO2 ~ trend()),
        trnd_ssn_w_wkday_dummy = TSLM(NO2 ~ trend() + day),
        trnd_ssn_w_wknd_dummy = TSLM(NO2 ~ trend() + weekend),
        trnd_ssn_wkly = TSLM(NO2 ~ trend() + sin(2*pi*step/7) + cos(2*pi*step/7)),
        trnd_ssn_wkly_mnthly = TSLM(NO2 ~ trend() + sin(2*pi*step/(365.25/12)) +
                                    cos(2*pi*step/(365.25/12)) +
                                    sin(2*pi*step/7) + cos(2*pi*step/7)),
        trnd_ssn_complex_trig = TSLM(NO2 ~ trend() + sin(2*pi*step/192) +
                                    cos(2*pi*step/192) + sin(2*pi*step/128) +
                                    cos(2*pi*step/128) + sin(2*pi*step/76.8) +
                                    cos(2*pi*step/76.8)+ sin(2*pi*step/64) +
                                    cos(2*pi*step/64)+ sin(2*pi*step/7) +
                                    cos(2*pi*step/7)+sin(2*pi*step/30) +
                                    cos(2*pi*step/30)+sin(2*pi*step/365) +
                                    cos(2*pi*step/365)),
        trnd_ssn_complex_trig_and_dummy = TSLM(NO2 ~ trend() + weekend +
                                    sin(2*pi*step/192) +
                                    cos(2*pi*step/192) + sin(2*pi*step/128) +
                                    cos(2*pi*step/128) + sin(2*pi*step/76.8) +
                                    cos(2*pi*step/76.8)+ sin(2*pi*step/64) +
                                    cos(2*pi*step/64)+ sin(2*pi*step/7) +
```

```
                                     cos(2*pi*step/7)+sin(2*pi*step/30) +
                                     cos(2*pi*step/30)+sin(2*pi*step/365) +
                                     cos(2*pi*step/365))
        )
```

```
# Inspect the trend only model
models %>% dplyr::select(trend) %>% report()
## Series: NO2
## Model: TSLM
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -88.444 -33.593   2.634  28.304 138.425
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 113.76664    4.51323   25.21   <2e-16 ***
## trend()       0.25902    0.02032   12.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 44.13 on 382 degrees of freedom
## Multiple R-squared: 0.2985,  Adjusted R-squared: 0.2966
## F-statistic: 162.5 on 1 and 382 DF, p-value: < 2.22e-16
```

The coefficient on trend is significant, and plotting the modeled trend line against the original data below seems to verify this. The significance of the trend will change when the seasonal components are modeled but it turns out to be significant in all cases.

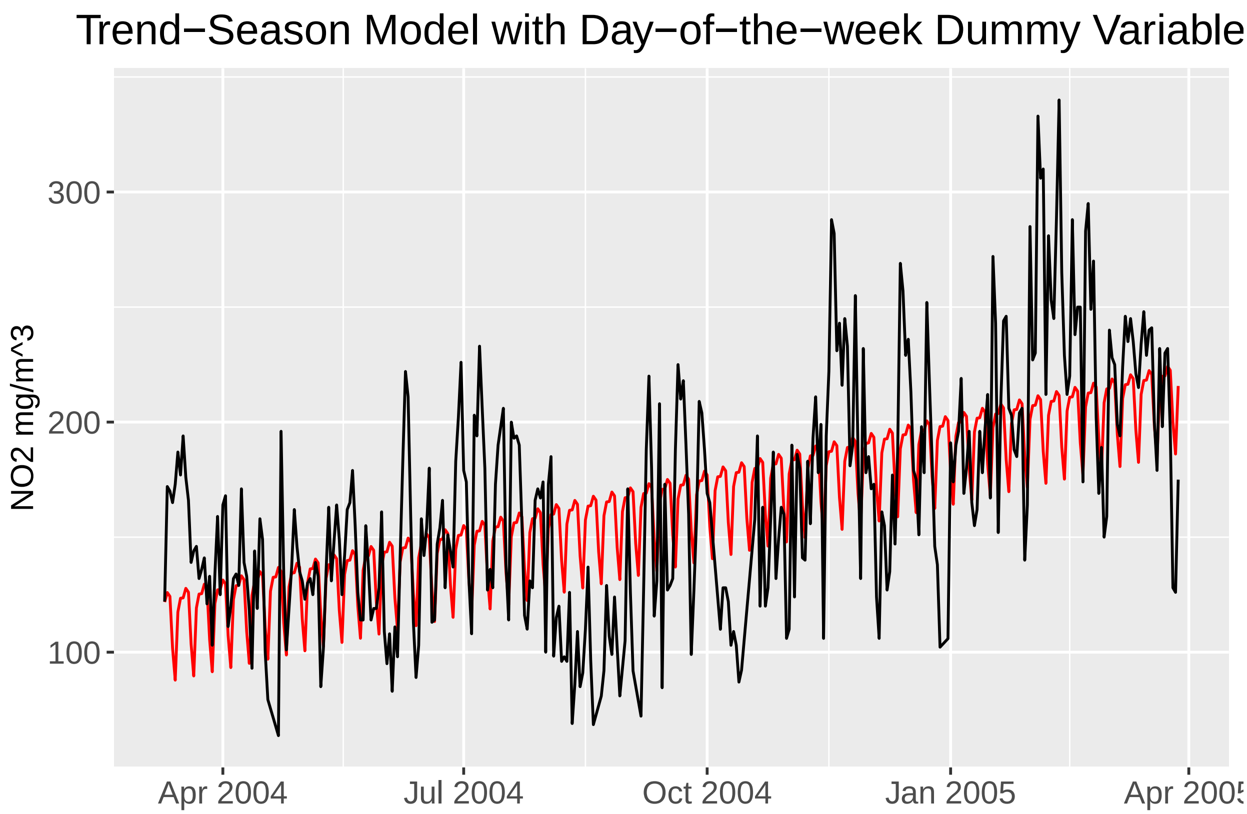
```
models %>% dplyr::select(trend) %>% fitted() %>%
  autoplot(color = 'red') + autolayer(dailyAQ_train) +
  ggtitle("Trend Model") + ylab("NO2 mg/m^3") + xlab("") + my_theme
## Plot variable not specified, automatically selected `.vars = .fitted`
## Plot variable not specified, automatically selected `.vars = NO2`
```



Next, we inspected the reports and plots of the trend season models that used day-of-the-weekend and weekend/not weekend dummy variables.

```
models %>% dplyr::select(trnd_ssn_w_wkday_dummy) %>% report()
## Series: NO2
## Model: TSLM
##
## Residuals:
##     Min      1Q Median      3Q     Max
## -99.28 -26.90   2.88  25.05 128.38
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 123.4759     6.8287  18.082  < 2e-16 ***
## trend()       0.2600     0.0195  13.333  < 2e-16 ***
## dayMon       -7.6160     8.0767  -0.943  0.34630
## daySat      -22.7403     8.0765  -2.816  0.00512 **
## daySun      -36.9244     8.0765  -4.572 6.57e-06 ***
## dayThu        1.9389     8.0765   0.240  0.81041
## dayTue       -1.8427     8.1138  -0.227  0.82046
## dayWed       -1.9503     8.0765  -0.241  0.80931
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.35 on 376 degrees of freedom
## Multiple R-squared: 0.3641,  Adjusted R-squared: 0.3523
```

```
## F-statistic: 30.75 on 7 and 376 DF, p-value: < 2.22e-16

models %>% dplyr::select(trnd_ssn_w_wkday_dummy) %>% fitted() %>%
  autoplot(color = 'red') + autolayer(dailyAQ_train) +
  ggtitle("Trend-Season Model with Day-of-the-week Dummy Variables") +
  ylab("NO2 mg/m^3") + xlab("") + my_theme
## Plot variable not specified, automatically selected `.vars = .fitted`
## Plot variable not specified, automatically selected `.vars = NO2`
```

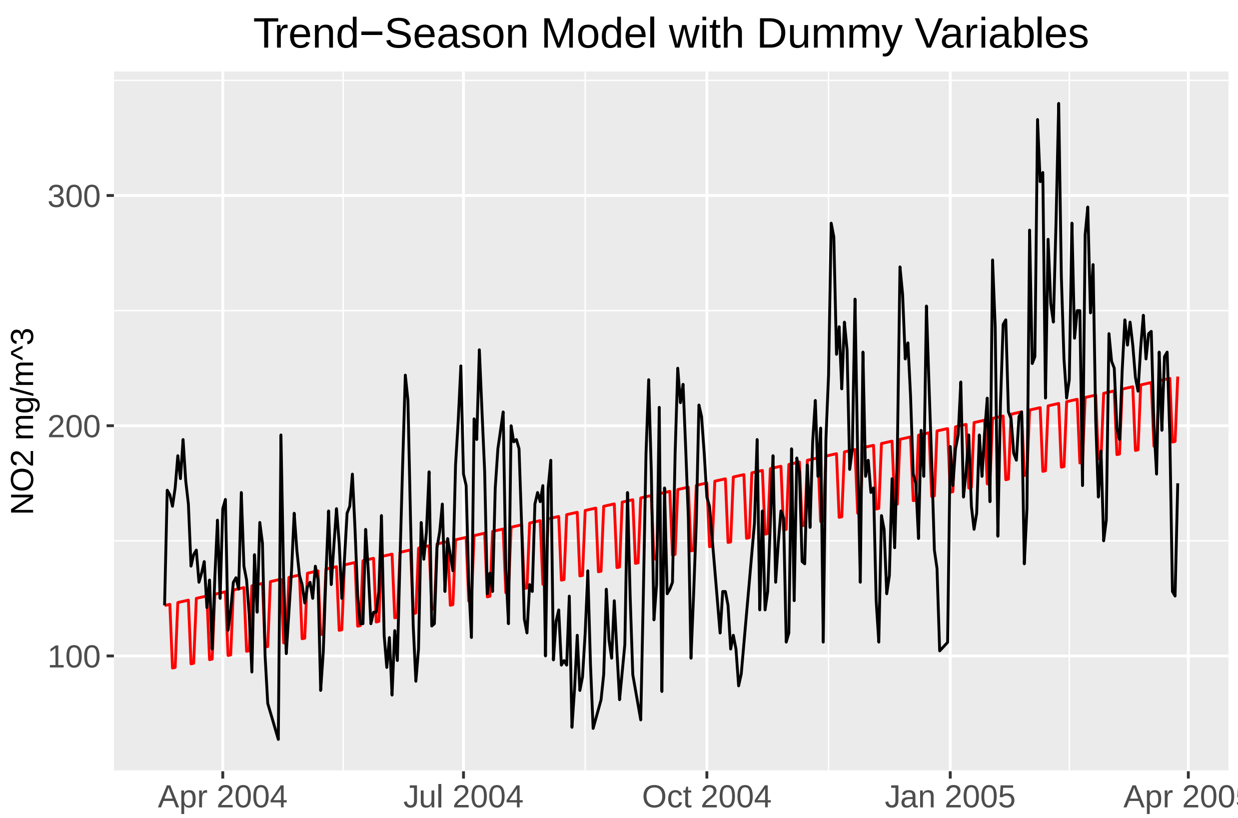


This model used a dummy variable for each day of the week. Friday is the base case. Only Saturday and Sunday were significantly different from the base case. NO2 concentrations on these days are lower. Perhaps traffic is heavier during the weekdays because of commuters. It is also notable that the trend term is still significant once the seasonality component has been accounted for.

The insignificance of the weekday variables led us to attempt to model the seasonal component with a single dummy variable indicating whether the date was on a weekday or weekend.

```
models %>% dplyr::select(trnd_ssn_w_wknd_dummy) %>% report()
## Series: NO2
## Model: TSLM
##
## Residuals:
##      Min       1Q  Median      3Q     Max
## -96.440 -27.579   2.535  26.753 130.327
##
## Coefficients:
```

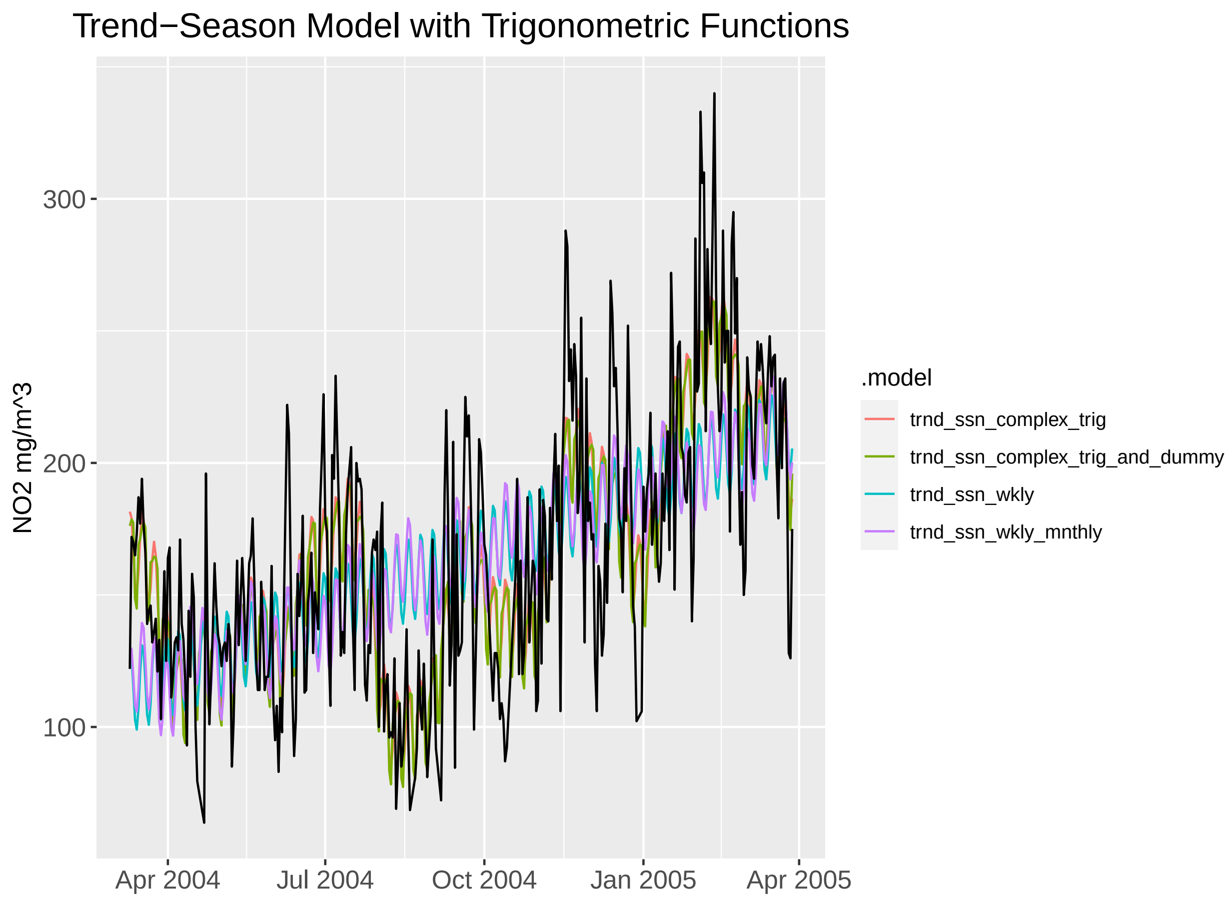
```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 121.64424    4.53395  26.830  < 2e-16 ***
## trend()       0.25967    0.01949  13.324  < 2e-16 ***
## weekendTRUE -27.93766    4.77846  -5.847 1.08e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 42.33 on 381 degrees of freedom
## Multiple R-squared: 0.3562,  Adjusted R-squared: 0.3529
## F-statistic: 105.4 on 2 and 381 DF, p-value: < 2.22e-16

models %>% dplyr::select(trnd_ssn_w_wknd_dummy) %>% fitted() %>%
  autoplot(color = 'red') + autolayer(dailyAQ_train) +
  ggtitle("Trend-Season Model with Dummy Variables") + ylab("NO2 mg/m^3") +
  xlab("") + my_theme
## Plot variable not specified, automatically selected `.vars = .fitted`
## Plot variable not specified, automatically selected `.vars = NO2`
```



All coefficients are significant indicating this is a strong candidate model.

Next we inspected the model reports and plot of the trend season models that relied on trigonometric functions. We show this all in a single graph and table to save space. Note that the model 'trnd_ssn_complex_trig_and_dummy' does in fact have a dummy variable indicating whether a day is a weekend or not. The table showing coefficients and p-values are sorted from highest p-value to lowest p-value.

```r
models %>% dplyr::select(trnd_ssn_wkly, trnd_ssn_wkly_mnthly,
                         trnd_ssn_complex_trig,
                         trnd_ssn_complex_trig_and_dummy) %>% coef() %>%
                         arrange(desc(p.value))
## # A tibble: 43 x 6
##    .model               term                estimate std.error statistic p.value
##    <chr>                <chr>                  <dbl>     <dbl>     <dbl>   <dbl>
##  1 trnd_ssn_complex_tr~ sin(2 * pi * step/~  0.00797      2.64   0.00302  0.998
##  2 trnd_ssn_complex_tr~ sin(2 * pi * step/~ -0.0453       2.59  -0.0175   0.986
##  3 trnd_ssn_wkly_mnthly cos(2 * pi * step/~ -0.931        3.05  -0.305    0.761
##  4 trnd_ssn_complex_tr~ cos(2 * pi * step/~ -2.09         3.11  -0.672    0.502
##  5 trnd_ssn_complex_tr~ sin(2 * pi * step/~  4.44         3.46   1.28     0.200
##  6 trnd_ssn_complex_tr~ sin(2 * pi * step/~  3.33         2.44   1.36     0.173
##  7 trnd_ssn_complex_tr~ sin(2 * pi * step/~  3.33         2.39   1.39     0.165
##  8 trnd_ssn_complex_tr~ sin(2 * pi * step/~  4.85         2.58   1.88     0.0604
##  9 trnd_ssn_wkly        cos(2 * pi * step/~  5.84         3.09   1.89     0.0597
## 10 trnd_ssn_complex_tr~ sin(2 * pi * step/~  4.81         2.53   1.90     0.0578
## # ... with 33 more rows

models %>%
  dplyr::select(trnd_ssn_wkly, trnd_ssn_wkly_mnthly,
  trnd_ssn_complex_trig, trnd_ssn_complex_trig_and_dummy) %>%
  fitted() %>% autoplot() +
  autolayer(dailyAQ_train) +
  ggtitle("Trend-Season Model with Trigonometric Functions") +
  ylab("NO2 mg/m^3") + xlab("") + my_theme
## Plot variable not specified, automatically selected `.vars = .fitted`
## Plot variable not specified, automatically selected `.vars = NO2`
```



Visually, the complex trig functions track the process very well though there are a few insignificant variables. Interestingly, the cosine function counterpart to the most insignificant variable of the trnd_ssn_complex_trig model (the sine function component of a 76.8 day season) has a p-value significant to the 0.001 level. This is not shown in the table but could easily be recreated from the attached .rmd.

To decide which is the best approach, we compared performance metrics across the candidate models with a trend and seasonal component.

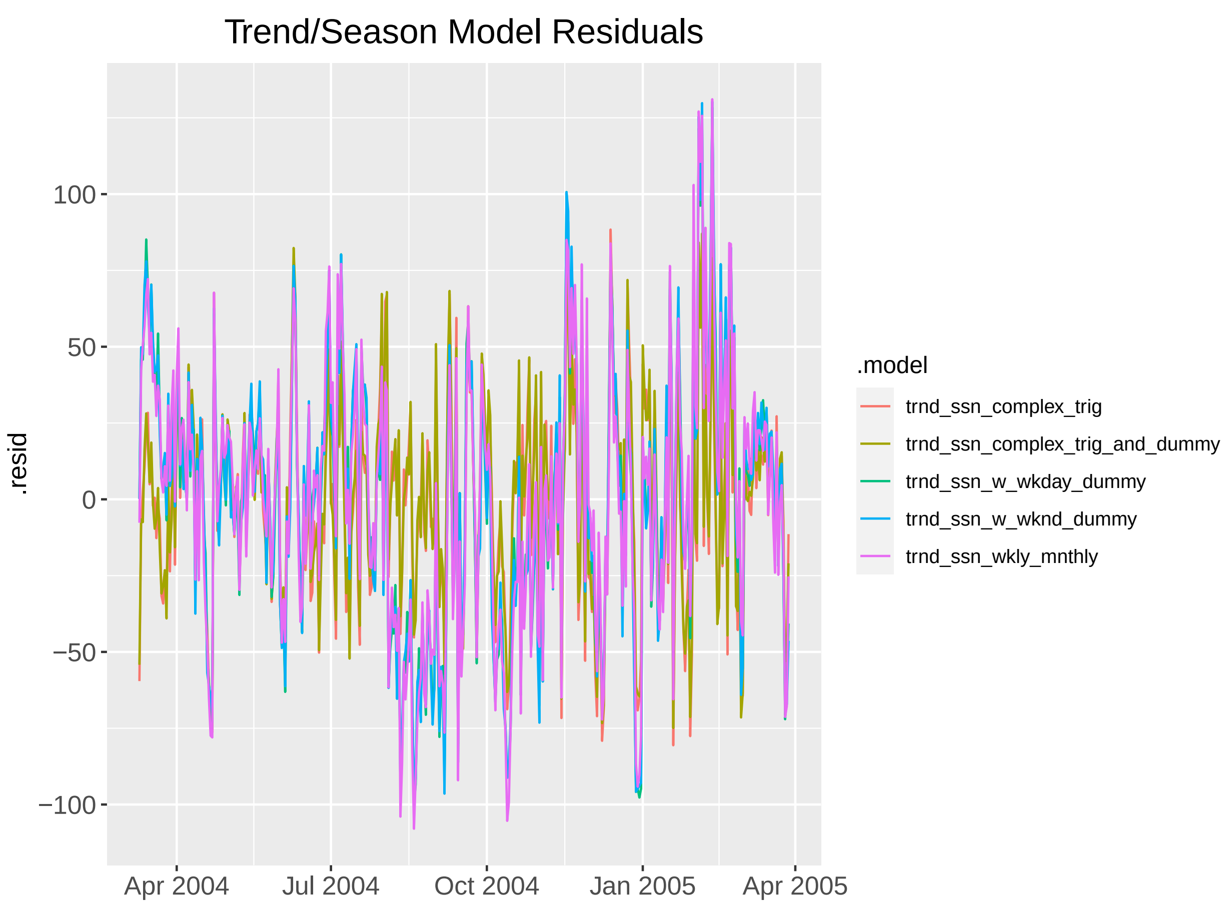
```
models %>%
  dplyr::select(trnd_ssn_wkly, trnd_ssn_wkly_mnthly, trnd_ssn_complex_trig,
  trnd_ssn_complex_trig_and_dummy, trnd_ssn_w_wkday_dummy, trnd_ssn_w_wknd_dummy) %>%
  report() %>% dplyr::select(.model, adj_r_squared, AIC, BIC, df.residual)
## Warning in report.mdl_df(.): Model reporting is only supported for individual
## models, so a glance will be shown. To see the report for a specific model, use
## `select()` and `filter()` to identify a single model.
## # A tibble: 6 x 5
##   .model                          adj_r_squared   AIC   BIC df.residual
##   <chr>                                   <dbl> <dbl> <dbl>       <int>
## 1 trnd_ssn_w_wkday_dummy                  0.352 2887. 2922.         376
## 2 trnd_ssn_w_wknd_dummy                   0.353 2882. 2897.         381
## 3 trnd_ssn_wkly                           0.340 2890. 2910.         380
## 4 trnd_ssn_wkly_mnthly                    0.350 2886. 2914.         378
## 5 trnd_ssn_complex_trig                   0.594 2715. 2782.         368
## 6 trnd_ssn_complex_trig_and_dummy         0.610 2701. 2772.         367
```

The trend/season models with the complex trigonometric functions clearly outperformed the other models, but we decided to include other model candidates in case these are overfit. Since the only difference between the trnd_ssn_wkly and trnd_ssn_wkly_mnthly model is the inclusion of a monthly trigonometric function and the latter performed better, we decided to exclude the trnd_ssn_wkly model from the rest of our analysis.

2.2 Modeling the residuals of best trend/season models

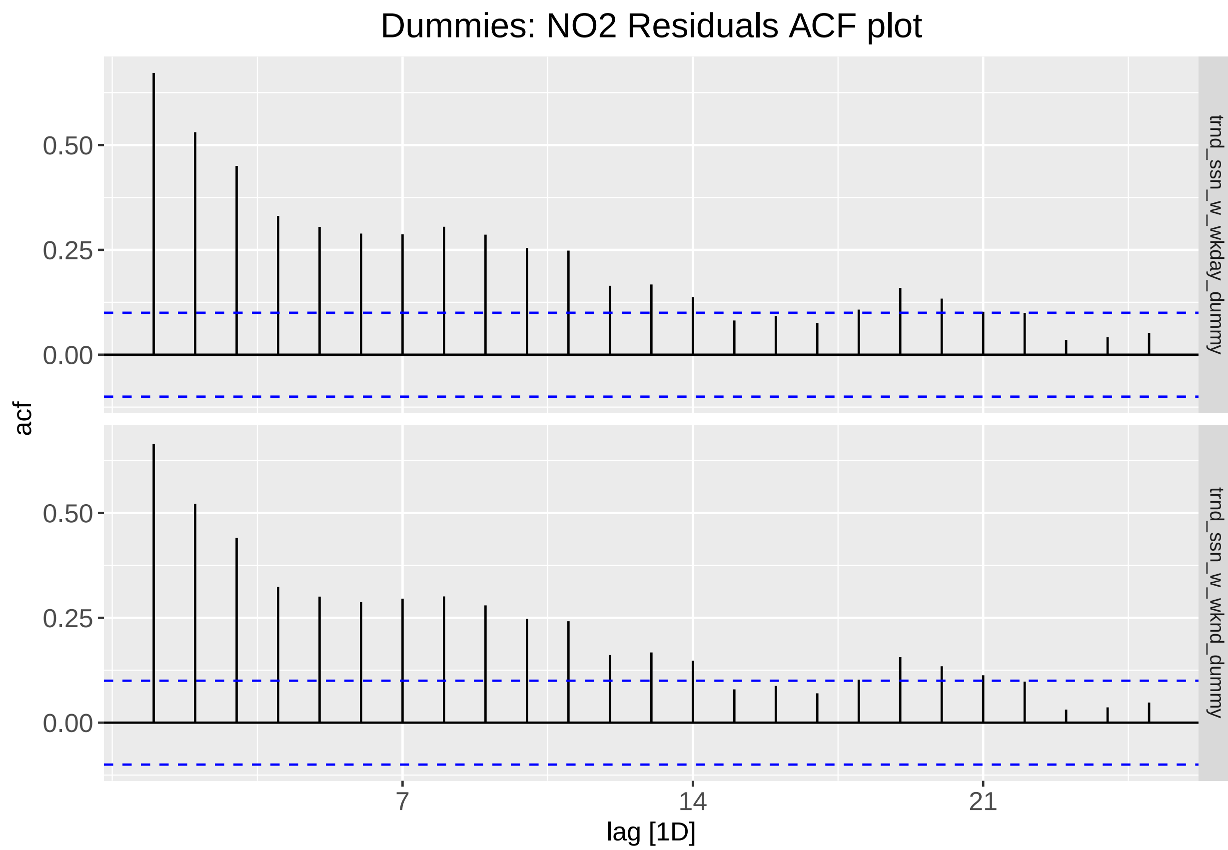
Next, we extracted and plotted the residuals of each candidate trend/season model to inspect for autoregressive properties.

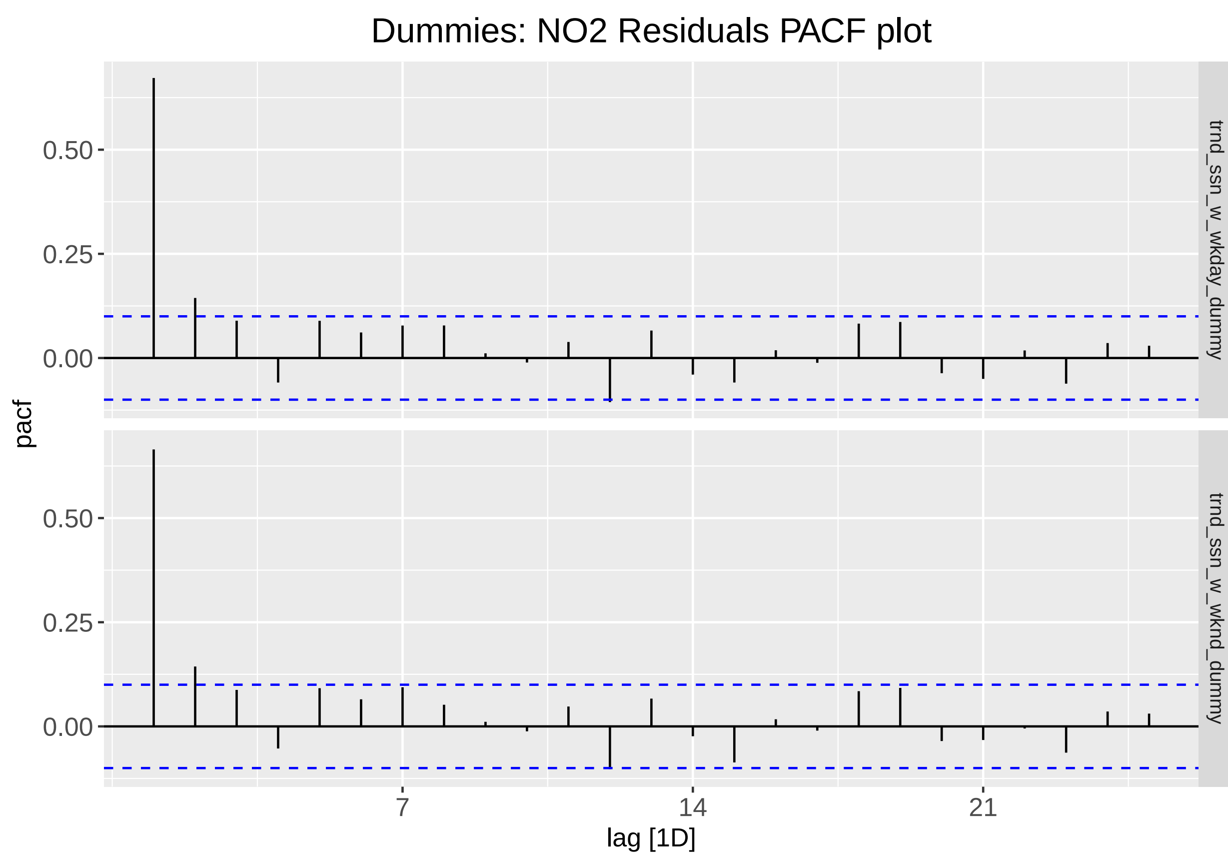
```
# Extract model residuals
data_resids <- models %>%
  dplyr::select(trnd_ssn_wkly_mnthly, trnd_ssn_w_wkday_dummy,
                trnd_ssn_w_wknd_dummy, trnd_ssn_complex_trig,
                trnd_ssn_complex_trig_and_dummy) %>%
                residuals()

# Plot the residuals
data_resids %>% autoplot(.resid) + ggtitle("Trend/Season Model Residuals") +
  xlab("") +  my_theme
```



There is clearly some autocorrelation in the residuals in all of the models because none of them look like random white noise. The next step was to inspect the ACF and PACF plots to decide how to model them. Because there are so many models, we look at the dummy variable models separate from the trigonometric models.

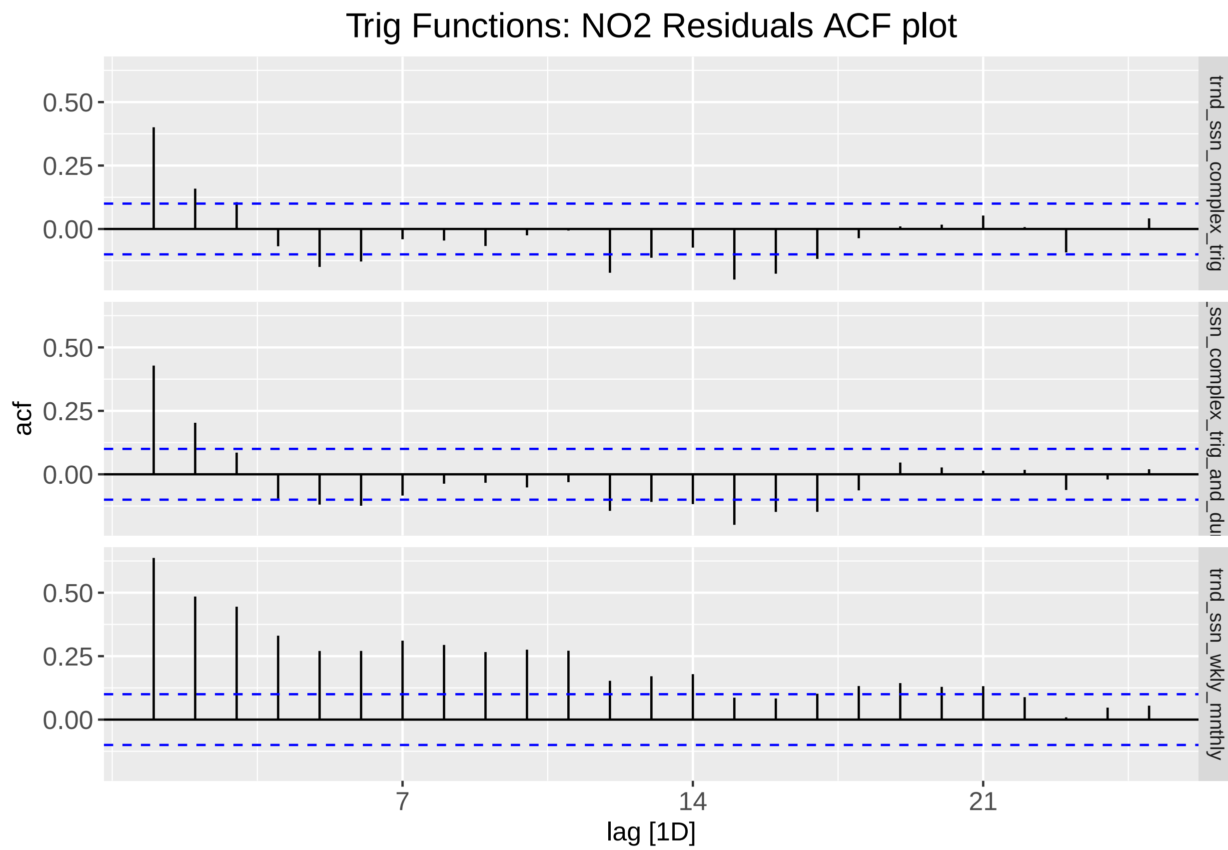
```
plt_resid_ACF <- data_resids %>%
  dplyr::filter(.model %in% c("trnd_ssn_w_wknd_dummy" ,"trnd_ssn_w_wkday_dummy")) %>%
  ACF(.resid) %>% autoplot() + ggtitle("Dummies: NO2 Residuals ACF plot") + my_theme

plt_resid_PACF <- data_resids %>%
  dplyr::filter(.model %in% c("trnd_ssn_w_wknd_dummy" ,"trnd_ssn_w_wkday_dummy")) %>%
  PACF(.resid) %>% autoplot() + ggtitle("Dummies: NO2 Residuals PACF plot") + my_theme

ggarrange(plt_resid_ACF,plt_resid_PACF,nrow=2,ncol=1)
```

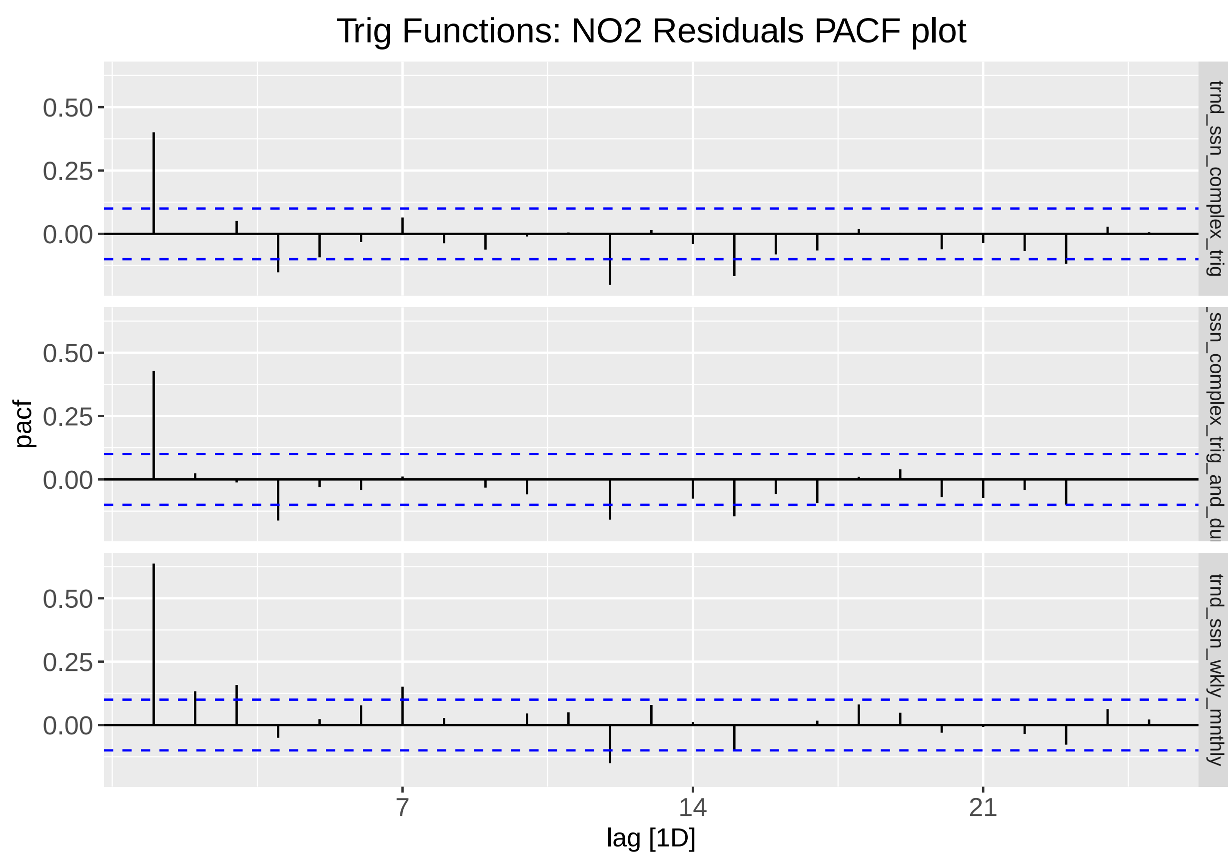

Dummies: NO2 Residuals ACF plot
acf
0.50
0.25
0.00
trnd_ssn_w_wkday_dummy
trnd_ssn_w_wknd_dummy
7
14
21
lag [1D]


Dummies: NO2 Residuals PACF plot
pacf
0.50
0.25
0.00
trnd_ssn_w_wkday_dummy
trnd_ssn_w_wknd_dummy
7
14
21
lag [1D]

Both dummy variable trend/season models exhibit similar behavior in the ACF and PACF. There is sinusoidal decay in the ACF of the residuals. The PACF could be interpreted as cutting off after lag 2 or as exhibiting sinusoidal behavior. This indicates that the residuals could be modeled as either AR(2) or as an ARMA process.

```
 plt_resid_ACF <- data_resids %>%
  dplyr::filter(.model %in% c("trnd_ssn_wkly_mnthly" ,"trnd_ssn_complex_trig",
                            "trnd_ssn_complex_trig_and_dummy")) %>%
                            ACF(.resid) %>%
                            autoplot() +
                            ggtitle("Trig Functions: NO2 Residuals ACF plot") +
                            my_theme

plt_resid_PACF <- data_resids %>%
  dplyr::filter(.model %in% c("trnd_ssn_wkly_mnthly" ,"trnd_ssn_complex_trig",
                              "trnd_ssn_complex_trig_and_dummy")) %>%
                            PACF(.resid) %>% autoplot() +
                            ggtitle("Trig Functions: NO2 Residuals PACF plot") +

ggarrange(plt_resid_ACF,plt_resid_PACF,nrow=2,ncol=1)
```


Trig Functions: NO2 Residuals ACF plot
acf
0.50
0.25
0.00
trnd_ssn_complex_trig
_ssn_complex_trig_and_du
trnd_ssn_wkly_mnthly
7
14
21
lag [1D]


Trig Functions: NO2 Residuals PACF plot
pacf
0.50
0.25
0.00
trnd_ssn_complex_trig
_ssn_complex_trig_and_du
trnd_ssn_wkly_mnthly
7
14
21
lag [1D]

The two trend/season models with more trigonometric functions (denoted 'complex') potentially show sinusoidal behavior in the ACF and the PACF and the PACF appears to cut off after 1 lag, although the second lag is technically barely above the significant threshold for the one that includes a weekend dummy variable. This indicates that it these should either be modeled as an AR(1) process or possibly an ARMA process if the PACF could be considered to be sinusoidal. The less complex trend/season model that only has monthly and weekly components looks more like the dummy variable models above with sinusoidal decay in the ACF and a PACF that either cuts off at lag 2, indicating AR(2), or a sinuisoidal PACF which would lead to modeling it as an ARMA process.

Next we performed autoregressive modeling of the residuals according to these hypotheses. Because of the structure of the tibble package, we had to apply the AR(1) and AR(2) models to all residual datasets. We also deployed an auto-selection process.

```
# Model the residuals
models_residuals <- data_resids %>%
  rename("model" = ".model", "resid" = ".resid") %>%
  model(auto.arima = ARIMA(resid ~ PDQ(0,0,0)),
        ar2 = ARIMA(resid ~ pdq(2, 0, 0) + PDQ(0,0,0)),
        ar1 = ARIMA(resid ~ pdq(1, 0, 0) + PDQ(0,0,0)))

# Show the pdq values from the auto.arima models
models_residuals %>% tidy() %>% filter(.model == 'auto.arima') %>% print(n = 3e3)
## # A tibble: 11 x 7
##    model                    .model    term  estimate std.error statistic  p.value
##    <chr>                    <chr>     <chr>    <dbl>     <dbl>     <dbl>    <dbl>
##  1 trnd_ssn_complex_trig    auto.ar~ ar1      0.403    0.0468      8.61 1.85e-16
##  2 trnd_ssn_complex_trig_a~ auto.ar~ ar1      0.431    0.0462      9.33 8.34e-19
##  3 trnd_ssn_w_wkday_dummy   auto.ar~ ar1      1.33     0.144       9.22 1.97e-18
##  4 trnd_ssn_w_wkday_dummy   auto.ar~ ar2     -0.375    0.115      -3.25 1.25e- 3
##  5 trnd_ssn_w_wkday_dummy   auto.ar~ ma1     -0.763    0.124      -6.17 1.74e- 9
##  6 trnd_ssn_w_wknd_dummy    auto.ar~ ar1      1.32     0.139       9.49 2.52e-19
##  7 trnd_ssn_w_wknd_dummy    auto.ar~ ar2     -0.369    0.111      -3.31 1.01e- 3
##  8 trnd_ssn_w_wknd_dummy    auto.ar~ ma1     -0.764    0.119      -6.41 4.39e-10
##  9 trnd_ssn_wkly_mnthly     auto.ar~ ar1      1.29     0.121      10.7  2.03e-23
## 10 trnd_ssn_wkly_mnthly     auto.ar~ ar2     -0.337    0.0972     -3.47 5.78e- 4
## 11 trnd_ssn_wkly_mnthly     auto.ar~ ma1     -0.772    0.101      -7.65 1.66e-13
```

The auto-selection process yielded ARMA(2, 1) models for the day-of-the-week and weekend/not weekend dummy variable models as well as the weekly and monthly trigonometric models. The auto-selection process yielded an AR(1) model for the complex_trig models as we hypothesized based on the ACF and PACF.

Next we report the AIC and BIC for each of the candidate residual models.

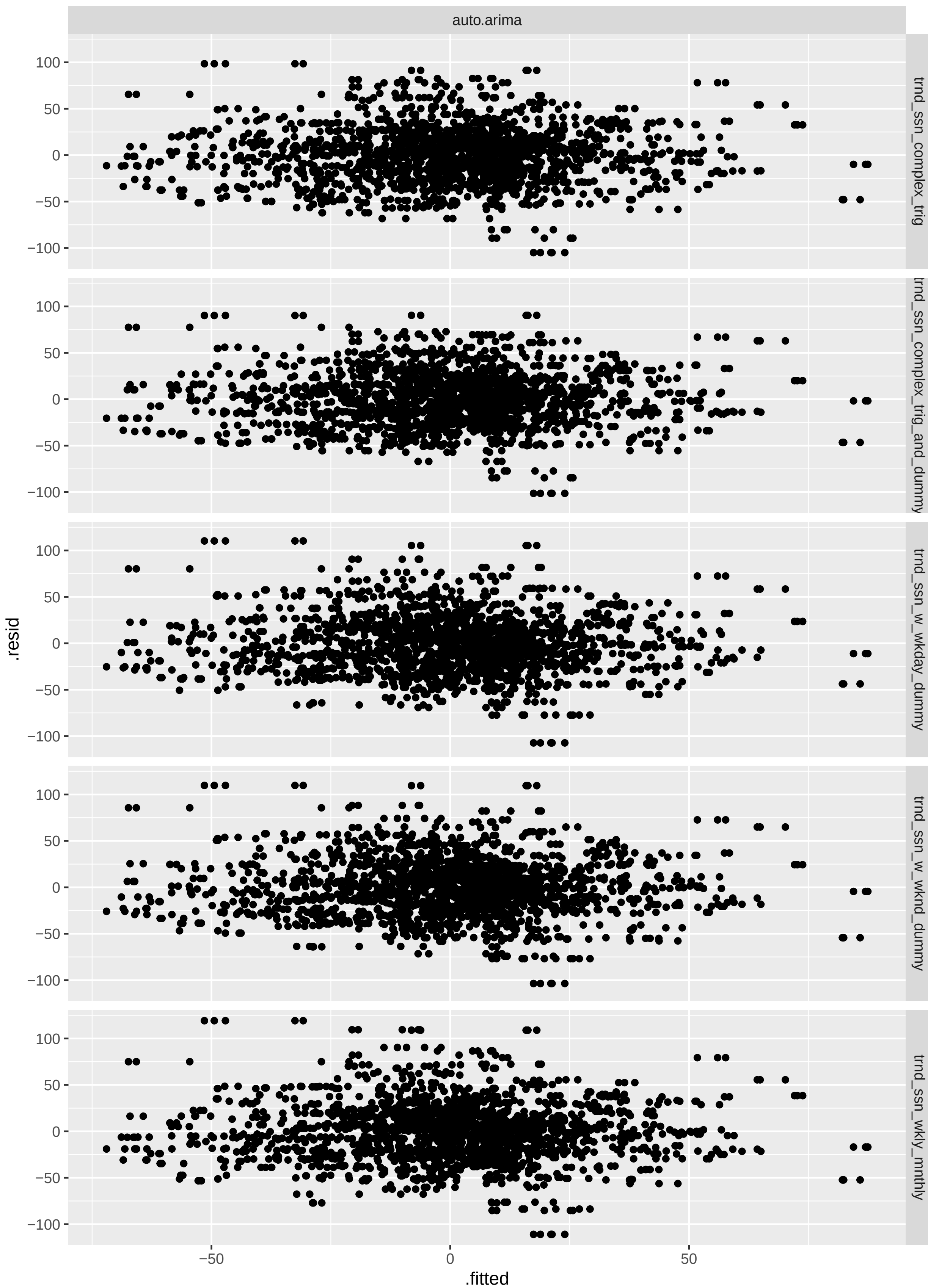
```
# Compare AIC and BIC across the models; sort by AIC
models_residuals %>% report() %>%
  dplyr::select(model, .model, AIC, BIC) %>% arrange(AIC) %>% print(n = 100)
## Warning in report.mdl_df(.): Model reporting is only supported for individual
## models, so a glance will be shown. To see the report for a specific model, use
## `select()` and `filter()` to identify a single model.
## # A tibble: 15 x 4
##    model                              .model       AIC   BIC
##    <chr>                              <chr>      <dbl> <dbl>
##  1 trnd_ssn_complex_trig_and_dummy auto.arima 3680. 3688.
##  2 trnd_ssn_complex_trig_and_dummy ar1        3680. 3688.
##  3 trnd_ssn_complex_trig_and_dummy ar2        3682. 3694.
```

```
##  4 trnd_ssn_complex_trig           auto.arima 3707. 3715.
##  5 trnd_ssn_complex_trig           ar1        3707. 3715.
##  6 trnd_ssn_complex_trig           ar2        3709. 3721.
##  7 trnd_ssn_w_wkday_dummy          auto.arima 3723. 3739.
##  8 trnd_ssn_w_wkday_dummy          ar2        3725. 3737.
##  9 trnd_ssn_w_wkday_dummy          ar1        3732. 3740.
## 10 trnd_ssn_w_wknd_dummy           auto.arima 3734. 3750.
## 11 trnd_ssn_w_wknd_dummy           ar2        3737. 3748.
## 12 trnd_ssn_w_wknd_dummy           ar1        3743. 3751.
## 13 trnd_ssn_wkly_mnthly            auto.arima 3754. 3770.
## 14 trnd_ssn_wkly_mnthly            ar2        3761. 3773.
## 15 trnd_ssn_wkly_mnthly            ar1        3766. 3774.
```

Based on AIC, the auto-selected ARIMA model performed best or tied for the best model for each set of residuals. BIC did not align exactly with AIC results. For both the dummy variable trend/season model residuals, the AR(2) model had the best score which was our hypothesis based on the AIC and BIC. The score improvement is likely negligable. For ease of visualization, we decided to continue with the rest of our analysis using only the auto-selected models.

Next we plotted residuals versus fitted and QQ plots of model residuals to test for the iid assumption.

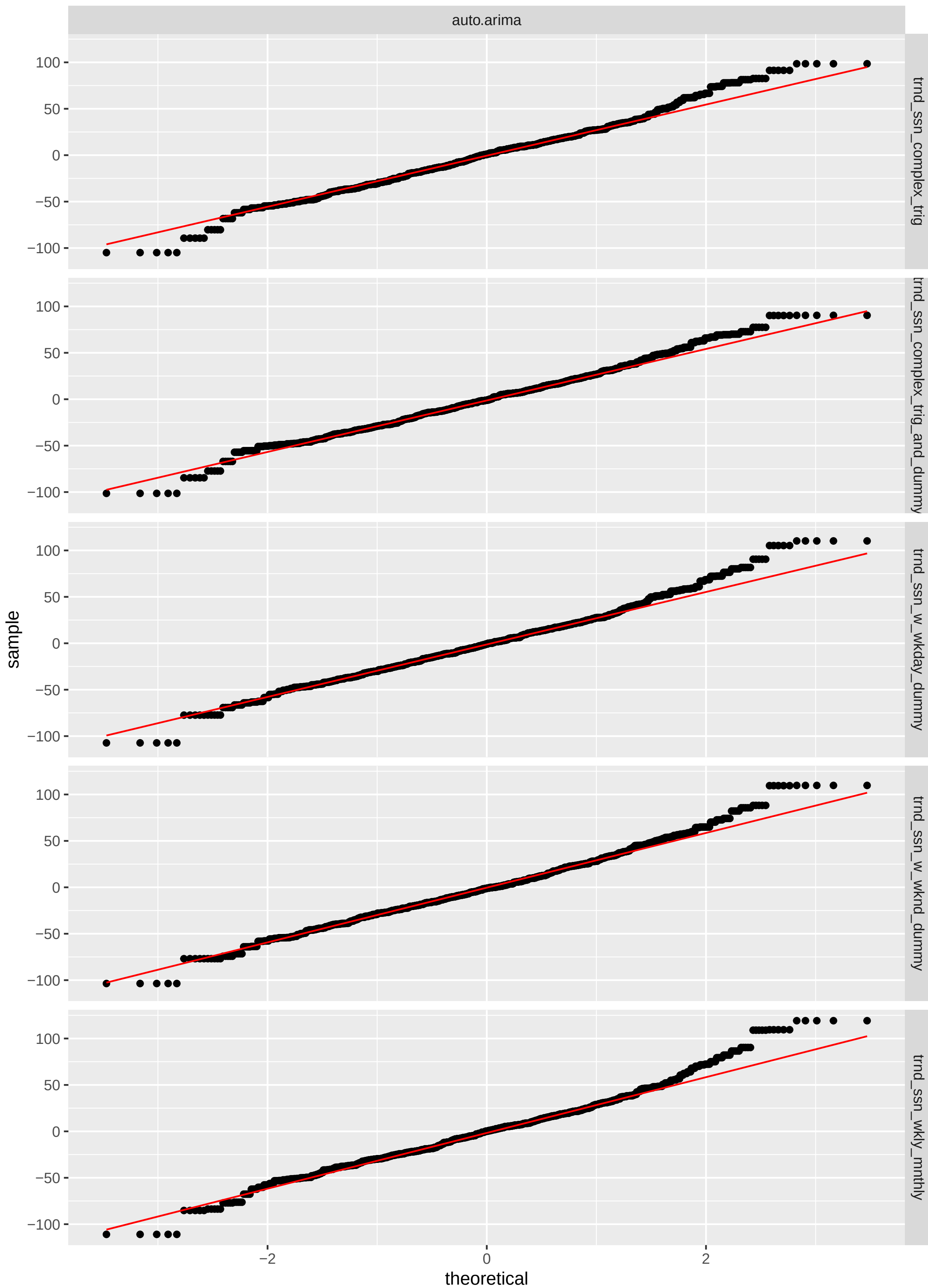
```r
# Choose the auto.arima model for the rest of the analysis
models_residuals <- models_residuals %>% dplyr::select(-ar2, -ar1)

# Plot residuals vs. fitted
models_residuals_fitted <-
  models_residuals %>% fitted() %>% as_tibble() %>% dplyr::select(Date, '.fitted')

models_residuals_resid <-
  models_residuals %>% residuals() %>% as_tibble()

model_residuals_fittedAndResiduals <-
  inner_join(models_residuals_resid, models_residuals_fitted, by = "Date")

ggplot(model_residuals_fittedAndResiduals, aes(.fitted, .resid)) +
  geom_point() + facet_grid(rows = vars(model), cols = vars(.model))
```



auto.arima
trnd_ssn_complex_trig
trnd_ssn_complex_trig_and_dummy
trnd_ssn_w_wkday_dummy
trnd_ssn_w_wknd_dummy
trnd_ssn_wkly_mnthly
.resid
.fitted
100
50
0
−50
−100
−50
0
50

There is no clear trend in any of the model residuals.

```
# Plot QQ plots
ggplot(model_residuals_fittedAndResiduals, aes(sample = .resid)) +
    stat_qq() + stat_qq_line(color = "red") +
    facet_grid(rows = vars(model), cols = vars(.model))
```


auto.arima
trnd_ssn_complex_trig
trnd_ssn_complex_trig_and_dummy
trnd_ssn_w_wkday_dummy
trnd_ssn_w_wknd_dummy
trnd_ssn_wkly_mnthly
sample
theoretical
100
50
0
−50
−100
−2
0
2

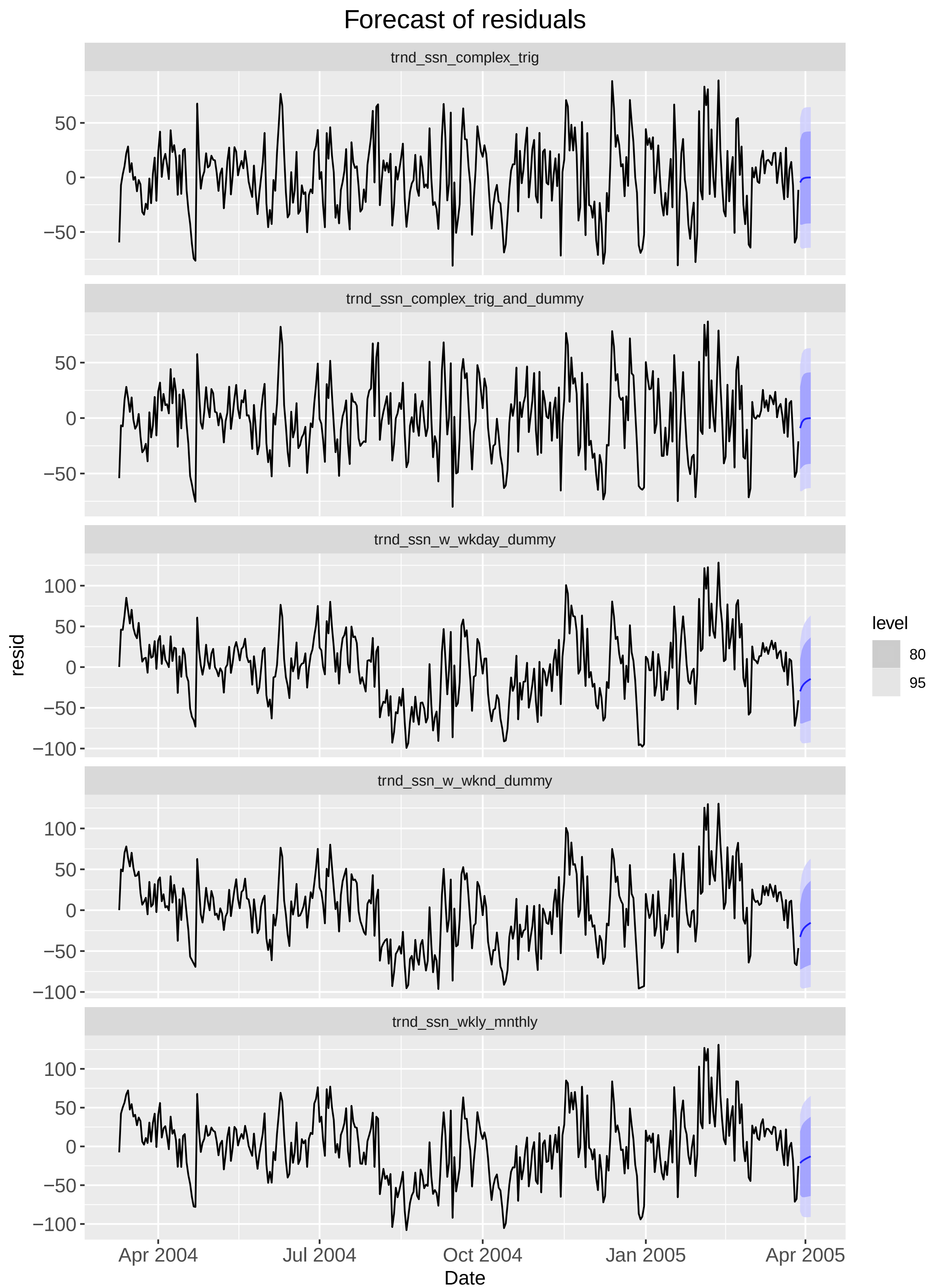
The residuals for each model all appear to exhibit similar patterns and are approximately gaussian with some fat-tail behavior.

The next step was to forecast and test each model against the test set.

3 Forecasting

We started by forecasting the residuals.

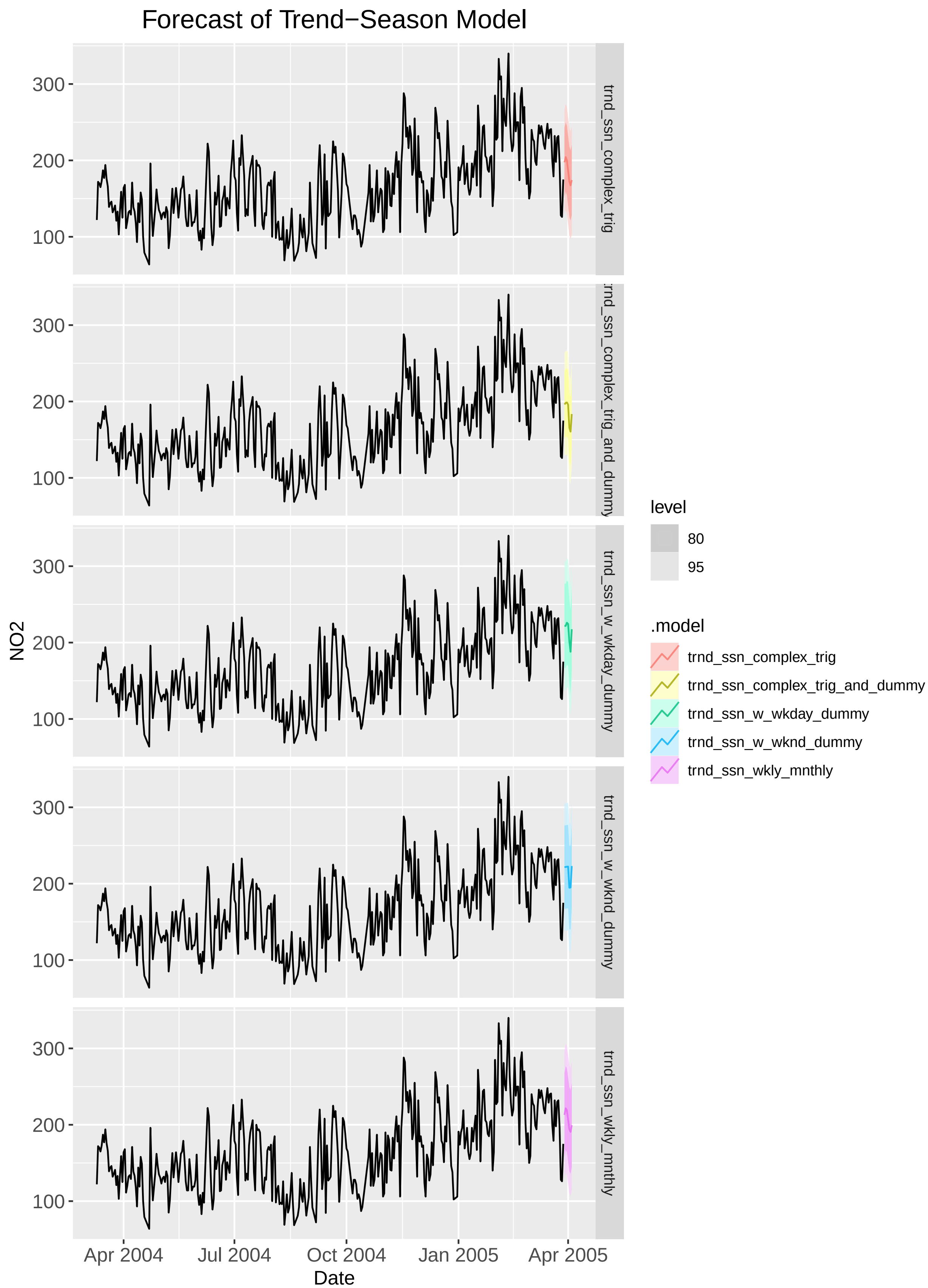
```
resid_summary <- models_residuals %>% augment()

resid_7day_forecast <- models_residuals %>%
  forecast(h = "1 week")

plt_resid_forecast <- resid_7day_forecast %>% feasts::autoplot(alpha = 0.8) +
  geom_line(data = resid_summary, aes(x = Date, y = resid)) +
  ggtitle("Forecast of residuals") + my_theme

plt_resid_forecast
```


Forecast of residuals
trnd_ssn_complex_trig
trnd_ssn_complex_trig_and_dummy
trnd_ssn_w_wkday_dummy
trnd_ssn_w_wknd_dummy
trnd_ssn_wkly_mnthly
resid
50
0
−50
100
−100
level
80
95
Apr 2004
Jul 2004
Oct 2004
Jan 2005
Apr 2005
Date

```r
resid_7day_forecast <- resid_7day_forecast %>% hilo(level = 95)
```

The forecast of residuals look reasonable and very similar as does the width of the confidence intervals. Next we forecasted the trend/season models.

```r
main_summary <- models %>% select(trnd_ssn_w_wknd_dummy,
                                  trnd_ssn_w_wkday_dummy,
                                  trnd_ssn_complex_trig_and_dummy,
                                  trnd_ssn_complex_trig,
                                  trnd_ssn_wkly_mnthly) %>% augment()

main_7day_forecast <- models %>%
  select(trnd_ssn_w_wknd_dummy, trnd_ssn_w_wkday_dummy,
         trnd_ssn_complex_trig_and_dummy,
         trnd_ssn_complex_trig, trnd_ssn_wkly_mnthly) %>%
         forecast(new_data = dailyAQ_test)

plt_main_forecast <- main_7day_forecast %>% feasts::autoplot(alpha = 0.8) +
        geom_line(data = main_summary, aes(x = Date, y = NO2)) +
        ggtitle("Forecast of Trend-Season Model") +
        facet_grid(rows = vars(.model)) +
        my_theme

plt_main_forecast
```


Forecast of Trend−Season Model
trnd_ssn_complex_trig
trnd_ssn_complex_trig_and_dummy
trnd_ssn_w_wkday_dummy
trnd_ssn_w_wknd_dummy
trnd_ssn_wkly_mnthly
NO2
100
200
300
Apr 2004
Jul 2004
Oct 2004
Jan 2005
Apr 2005
Date
level
80
95
.model
trnd_ssn_complex_trig
trnd_ssn_complex_trig_and_dummy
trnd_ssn_w_wkday_dummy
trnd_ssn_w_wknd_dummy
trnd_ssn_wkly_mnthly

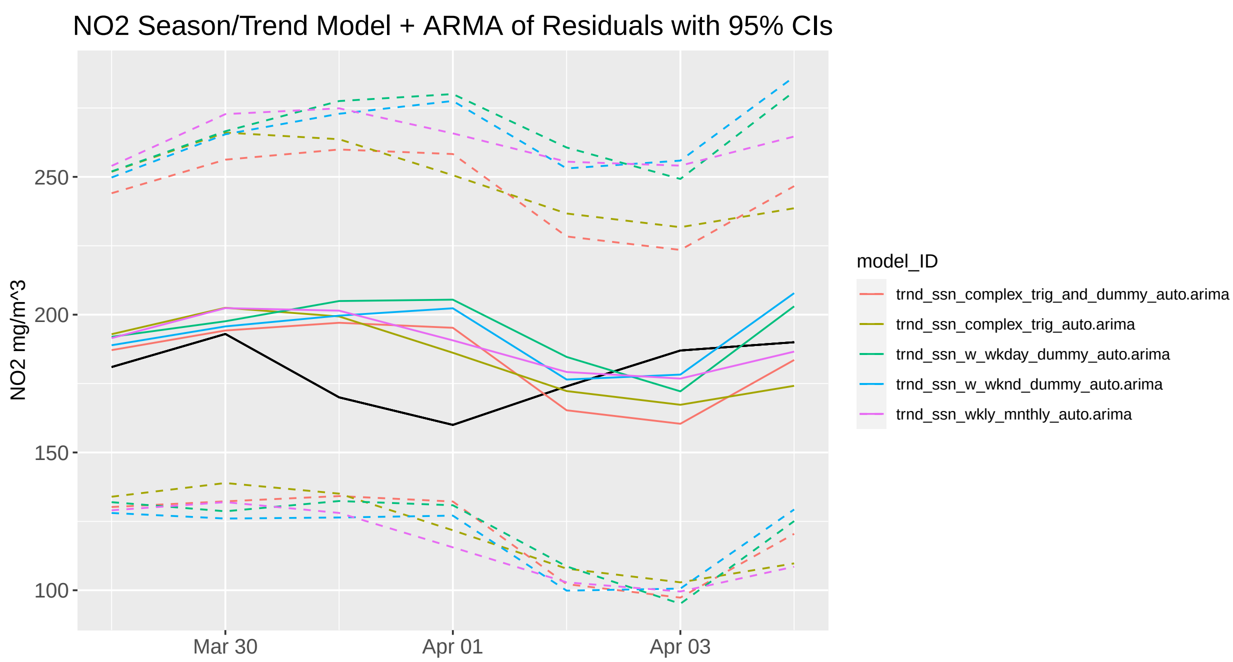
The trend/season forecasts also look similar with similar confidence intervals. The next step was to combine these models to form the full forecast and to evaluate performance. We show the mean squared error for each model after the plot below.

3.1 Combining Season/trend and residuals forecasts and computing MSE

```
full_model_summary <- resid_summary %>%
            left_join(main_summary,
            by = c("Date" = "Date", "model" = ".model"),
            suffix = c(".resid", ".full"))

full_forecast <- left_join(resid_7day_forecast, main_7day_forecast,
            by = c("model" = ".model", "Date" = "Date"),
            suffix = c(".resid", ".full")) %>%
            mutate(pt_pred = .mean.resid + .mean.full)

test_data <- dailyAQ_test %>%
            dplyr::select(Date, "NO2") %>%
            rename("NO2_Actual" = "NO2")

full_forecast <- full_forecast %>% inner_join(test_data, by = ("Date" = "Date"))

forecast_MSE <- full_forecast %>% as_tibble() %>% group_by(model, .model) %>%
            mutate(sqrd_error = (pt_pred - NO2_Actual)^2) %>%
            summarize(MSE = mean(sqrd_error))
## `summarise()` regrouping output by 'model' (override with `.groups` argument)
```

3.2 Ploting forecasts and MSE

```
full_forecast <- full_forecast %>% unpack_hilo(cols = "95%") %>%
  mutate(`95%_upper` = .mean.full + `95%_upper`) %>%
  mutate(`95%_lower` = .mean.full + `95%_lower`) %>%
  mutate(model_ID = paste(model, .model, sep = "_"))

plot_forecasts <- ggplot(data = full_forecast, aes(group = model_ID, color = model_ID)) +
  geom_line(aes(x=Date,y=NO2_Actual), color = "black") +
  geom_line(aes(x=Date,y=pt_pred)) +
  geom_line(aes(x=Date,y=`95%_lower`),linetype="dashed") +
  geom_line(aes(x=Date,y=`95%_upper`),linetype="dashed") +
  xlab("") + ylab("NO2 mg/m^3") +
  ggtitle("NO2 Season/Trend Model + ARMA of Residuals with 95% CIs") + my_theme

plot_forecasts
```




```
forecast_MSE
## # A tibble: 5 x 3
## # Groups:   model [5]
##   model                              .model       MSE
##   <chr>                              <chr>      <dbl>
## 1 trnd_ssn_complex_trig              auto.arima  346.
## 2 trnd_ssn_complex_trig_and_dummy    auto.arima  405.
## 3 trnd_ssn_w_wkday_dummy             auto.arima  562.
## 4 trnd_ssn_w_wknd_dummy              auto.arima  448.
## 5 trnd_ssn_wkly_mnthly               auto.arima  324.
```

Despite being significantly outperformed based on AIC, BIC, adjust R^2, and residuals, the model with the trigonometric functions representing weekly and monthly cycles performed best on the test dataset. That is the model we have selected. This indicates that the models with many trigonometric functions were overfitted.

Next we used the best model to simulate the next year of data.

4 Simulation

4.1 Simulating one year proceeding our observations

We started by forecasting the residuals.

```
resid_1year_simulation <- models_residuals %>% dplyr::select(auto.arima) %>%
  filter(model == "trnd_ssn_wkly_mnthly") %>%
  forecast(h = "1 year") # %>% hilo(level = 95)

resid_1year_simulation <- models_residuals %>% dplyr::select(auto.arima) %>%
  filter(model == "trnd_ssn_wkly_mnthly") %>% generate(h = "1 year", seed = 1)
```

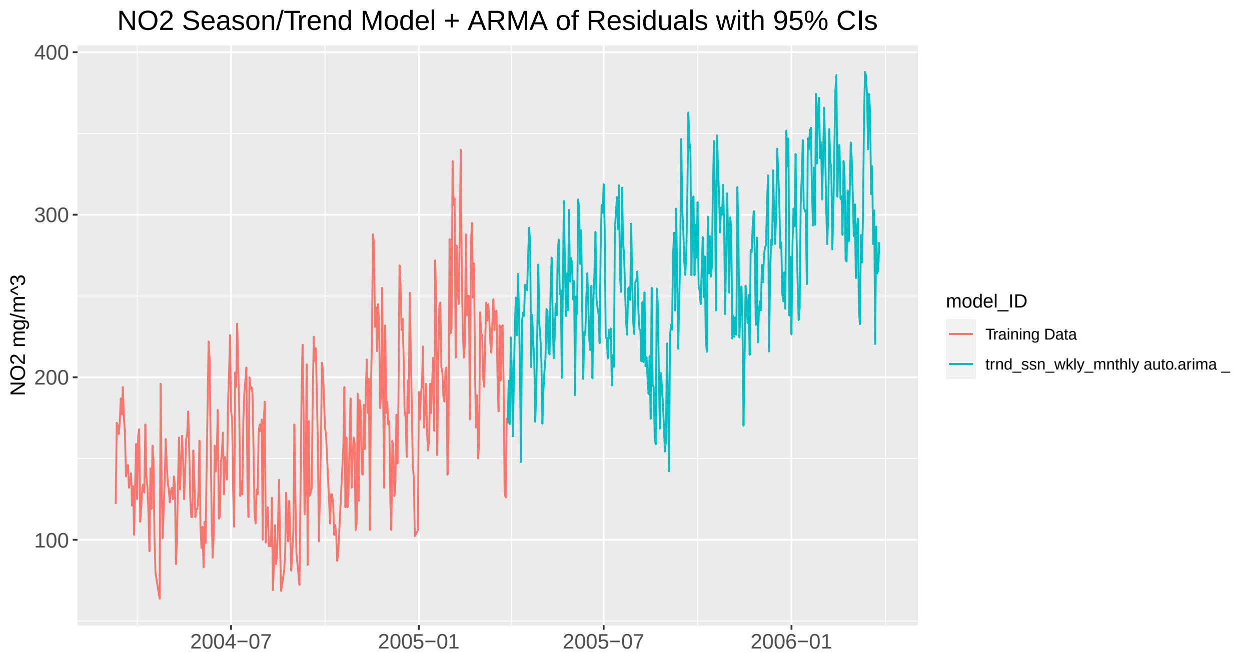
Next we forecasted the best trend/season model.

```
vec_simulation_dates <- seq(ymd('2005-03-29'),ymd('2006-03-29'),by='days')

original_steps <- length(dailyAQ_train$Date)
sim_steps <- length(vec_simulation_dates)

simulation_dates <- as_tibble() %>% as_tibble(.rows = length(vec_simulation_dates)) %>%
  bind_cols(vec_simulation_dates) %>% rename('Date' = '...1') %>%
  mutate(NO2 = -999) %>%
  mutate(step = (original_steps+1):(original_steps + sim_steps)) %>%
  mutate(day = as.factor(as.character(lubridate::wday(Date, label  = TRUE)))) %>%
  mutate(weekend = isWeekend(Date)) %>% as_tsibble(index = Date)
## Warning: The `x` argument of `as_tibble()` can't be missing as of lifecycle 3.0.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
## New names:
## * NA -> ...1


main_1year_forecast <- models %>%
  select(trnd_ssn_wkly_mnthly) %>%
  forecast(new_data = simulation_dates)
```

Finally, we combined them into a single simulation.

```
full_simulation <- left_join(resid_1year_simulation, main_1year_forecast,
                             by = c("model" = ".model", "Date" = "Date"),
                             suffix = c(".resid", ".full")) %>%
                             mutate(pt_pred = .sim + .mean)  %>%
                             mutate(model_ID = paste(model, .model, "_"))
```

```
dailyAQ_train <- dailyAQ_train %>% mutate(model_ID = "Training Data")

plot_simulation <- ggplot(data = full_simulation, aes(group = model_ID,
  color = model_ID)) +  geom_line(aes(x=Date,y=pt_pred)) +
  geom_line(data = dailyAQ_train, aes(x = Date, y = NO2)) +
  xlab("") + ylab("NO2 mg/m^3") +
  ggtitle("NO2 Season/Trend Model + ARMA of Residuals with 95% CIs") + my_theme

plot_simulation
```



The final simulation looks reasonable, so we are comfortable reporting this model as our final product. Next we compared the coefficient on the trend terms from the trend/season model built from the observations and a trend/season model built from the simulation.

4.2 Comparing trend/season models built from observations vs. simulation

```
# Isolate the final trend/season model used for simulation
models %>% dplyr::select(trnd_ssn_wkly_mnthly)
## # A mable: 1 x 1
##   trnd_ssn_wkly_mnthly
##                <model>
## 1               <TSLM>

# Create a trend/season model of the simulation
simulation_model <- full_simulation %>% dplyr::select(-model, -.model, -.rep) %>%
  model(sim_trnd_ssn_wkly_mnthly = TSLM(pt_pred ~ trend() + sin(2*pi*step/(365.25/12)) +
  cos(2*pi*step/(365.25/12)) + sin(2*pi*step/7) + cos(2*pi*step/7)))

# Show the model coefficients and p-values
models %>% dplyr::select(trnd_ssn_wkly_mnthly) %>% coef()
## # A tibble: 6 x 6
##   .model             term                   estimate std.error statistic  p.value
##   <chr>              <chr>                     <dbl>     <dbl>     <dbl>    <dbl>
## 1 trnd_ssn_wkly_mn~ (Intercept)               113.       4.34      26.1   1.33e-86
## 2 trnd_ssn_wkly_mn~ trend()                     0.261    0.0195    13.4   1.42e-33
## 3 trnd_ssn_wkly_mn~ sin(2 * pi * step/(36~     8.77     3.07       2.86  4.54e- 3
## 4 trnd_ssn_wkly_mn~ cos(2 * pi * step/(36~    -0.931    3.05      -0.305 7.61e- 1
## 5 trnd_ssn_wkly_mn~ sin(2 * pi * step/7)      14.9      3.06       4.87  1.67e- 6
## 6 trnd_ssn_wkly_mn~ cos(2 * pi * step/7)       5.84     3.06       1.90  5.76e- 2
```

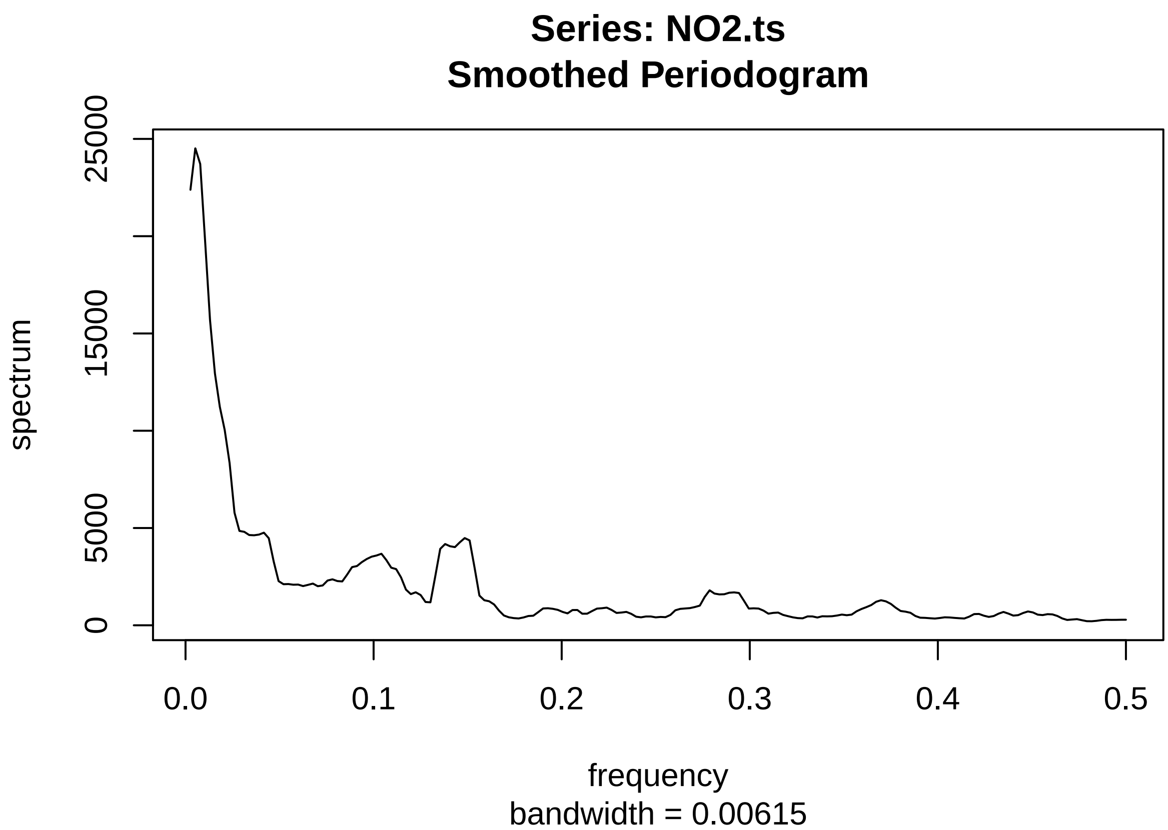
```r
simulation_model %>% coef()
## # A tibble: 6 x 6
##   .model              term                   estimate std.error statistic   p.value
##   <chr>               <chr>                     <dbl>     <dbl>     <dbl>     <dbl>
## 1 sim_trnd_ssn_wkly~ (Intercept)            219.        4.08       53.8  8.69e-174
## 2 sim_trnd_ssn_wkly~ trend()                  0.262     0.0193     13.6  3.60e- 34
## 3 sim_trnd_ssn_wkly~ sin(2 * pi * step/(~     8.74      2.87        3.04 2.53e-  3
## 4 sim_trnd_ssn_wkly~ cos(2 * pi * step/(~    -6.16      2.87       -2.14 3.28e-  2
## 5 sim_trnd_ssn_wkly~ sin(2 * pi * step/7)    12.3       2.88        4.27 2.46e-  5
## 6 sim_trnd_ssn_wkly~ cos(2 * pi * step/7)     4.37      2.87        1.53 1.28e-  1

# Extract the trend coefficient
main_coefs_trend <- models %>% dplyr::select(trnd_ssn_wkly_mnthly) %>% coef() %>%
  filter(term == "trend()") %>% pull(estimate)
sim_coefs_trend <- simulation_model %>% coef() %>% filter(term == "trend()") %>%
  pull(estimate)


# Calculate and report the percent difference between the trend coefficients of
# the model built from observed data and the model built from the simulation
perc_dif <- (sim_coefs_trend - main_coefs_trend)/main_coefs_trend * 100

perc_dif
## [1] 0.5312944
```

The trend coefficient of the model built from the simulation is 0.53% higher than the trend coefficient of the model built from the observations. This is evidence that the simulation is sufficiently matching observed patterns.

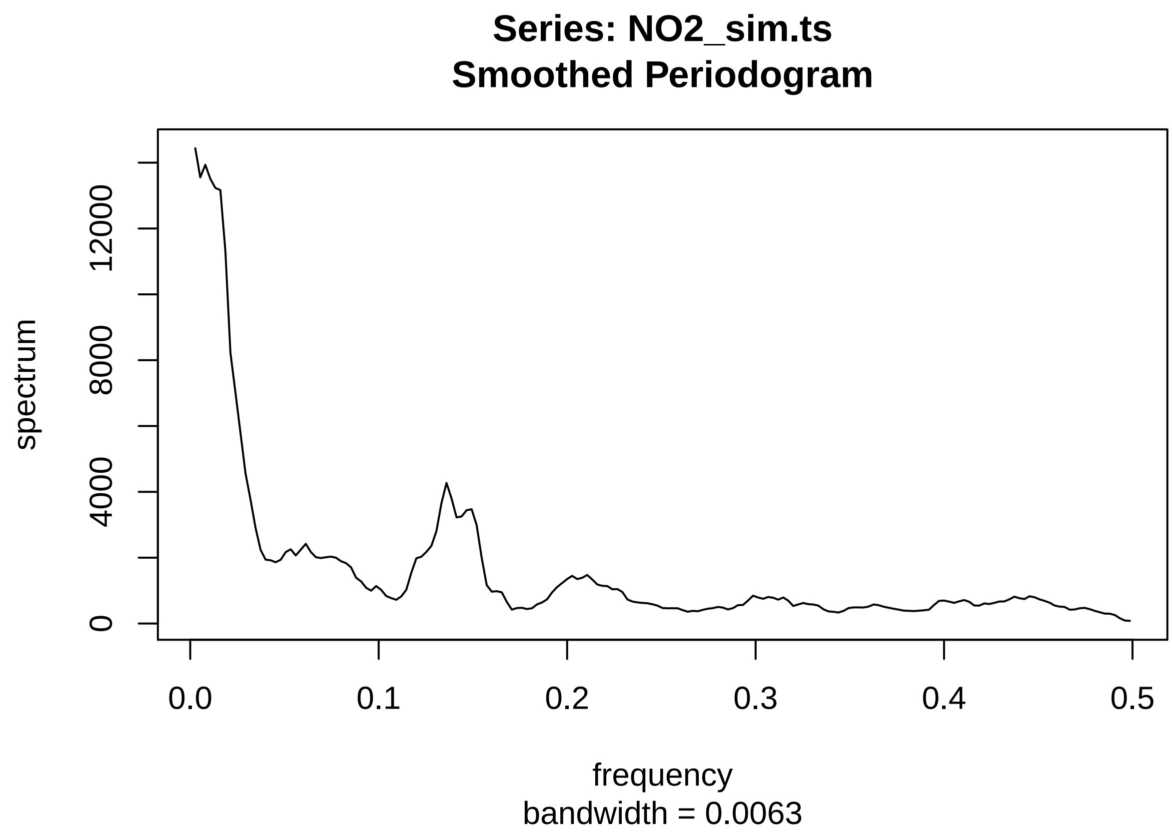
Next we inspected the periodogram to verify that the simulation reproduces the seasonal patterns of the observed data.

4.3 Verifying simulation reproduces seasonality

```r
NO2_sim.ts <- ts(full_simulation %>% as_tibble() %>% dplyr::select(pt_pred))

pg.NO2 <- spec.pgram(NO2.ts,spans=9,demean=T,log='no')
```




```
pg.NO2_sim <- spec.pgram(NO2_sim.ts,spans=9,demean=T,log='no')
```



The periodograms built from the observed data and the simulation data look very similar which further verifies the simulation. Next we compared sample statistics for the observed data and the simulated data.

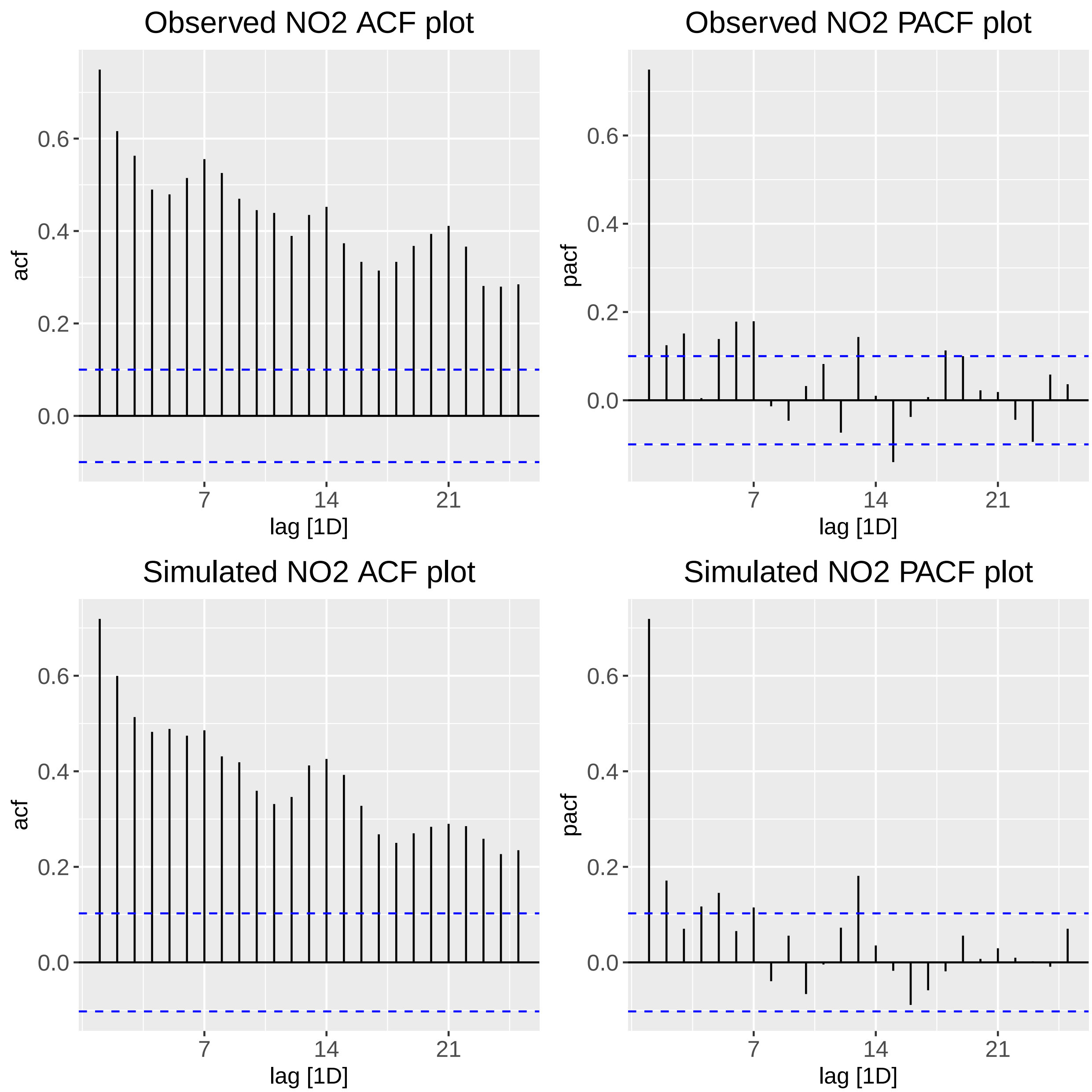
```
obs_mean <- mean(dailyAQ_train$NO2)
obs_var <- var(dailyAQ_train$NO2)

sim_mean <- mean(full_simulation$pt_pred)
sim_var <- var(full_simulation$pt_pred)

obs_mean_percDiff <- (sim_mean - obs_mean)/obs_mean*100
obs_var_percDiff <- (sim_var - obs_var)/obs_var*100

obs_mean_percDiff
## [1] 63.2018
obs_var_percDiff
## [1] -12.94752
```

The percent difference between the observed mean and simulated mean is 63.2% higher which is unsurprising since the simulation assumes the positive trend observed in the observations continues into the simulation period. The variance is 12.9% lower in the simulation which is indicative of a possible limitation of the simulation: it might not capture extremes well.

Finally, to verify that the simulation captures the autocorrelation structure of the observations, we compared the ACF and PACF plots.

4.4 Verifying simulation reproduces autocorrelation structure

```r
# Create plots of observed data
plt_NO2_obs_ACF <- dailyAQ_train %>%
  ACF(NO2) %>% autoplot() + ggtitle("Observed NO2 ACF plot") + my_theme

plt_NO2_obs_PACF <- dailyAQ_train %>%
  PACF(NO2) %>% autoplot() + ggtitle("Observed NO2 PACF plot") + my_theme

# Create plots of simulated data
plt_NO2_sim_ACF <- full_simulation %>%
  ACF(pt_pred) %>% autoplot() + ggtitle("Simulated NO2 ACF plot") + my_theme

plt_NO2_sim_PACF <- full_simulation %>%
  PACF(pt_pred) %>% autoplot() + ggtitle("Simulated NO2 PACF plot") + my_theme

ggarrange(plt_NO2_obs_ACF,plt_NO2_obs_PACF, plt_NO2_sim_ACF, plt_NO2_sim_PACF,nrow=2,
  ncol=2)
```



The ACF plot of both the observed data and the simulated data exhibit sinusoidal decay with similar levels of significance. The PACF plots show a steep drop off after the 1st lag with a few significant spikes and posible sinusoidal behavior. The similarities of these plots also serves to verify our simulation model.