Time Series Analysis for the Web Traffic of Public Figures

Sid Van Dam - jc2493

Sreeraj Yeduru - uy8958

Aakar Kale - rm8233

Sharmada Krishna G - wz9376

Summary

This time series analysis and forecast is concerning celebrity popularity. In the current times, the interest of a demographic can rapidly shift from one person to another in an instant. From the release of something like a movie or song, to a controversy, to being overshadowed by another, the current interest in a celebrity or public figure is something that is hard to capture outside of the immediate moment. By analyzing the Wikipedia page traffic of a number of hand selected public figures, this time series analysis attempts to forecast the general public's interest in these celebrities. The forecasting employs the use of more basic models, such as the naïve and moving average, to more advanced models like regression models, two level forecasting, and ARIMA. Through these multiple forecasting techniques, the models will visualize if there are patterns to a celebrity's growth or decline in the public eye. By using these patterns, should they exist, future growth or decline can be forecasted.

From the visualization on our datasets, we observed that each person we analyzed had a spike in their daily page traffic on at least one occasion, signaling that some event most likely occurred that caused people to view their Wikipedia article. The data itself had little autocorrelation, but there was trend and seasonality observed in the data.

The best performing model was the regression model with seasonality and quadratic trend. By using RMSE and MAPE as metrics, we can evaluate the models against one another, and the regression model with seasonality and quadratic trend had the best performing RMSE and MAPE. In addition, the forecast from the model was most able to accurately predict the future data compared to the other models employed.

Introduction

The dataset comes from a Kaggle dataset that provides a set of details concerning an exceptionally large number of Wikipedia articles. These articles are all different public figures, and such details include things like what kind of device was used to access these articles and the amount of traffic these articles received. The data provided covers the span of a few years and provides daily traffic for each public figure.

By being able to forecast the growth or decline of a public figure, certain agencies can make better informed decisions on who would be a good investment. For example, a casting agency can observe that a certain actor is gaining popularity in the public eye, which would make them a good pick for an upcoming movie. Alternatively, a music label can see that a music artist they were considering has had a steady decline, helping the label see that maybe that artist is not the best choice.

The type of public figures that were provided ranged from political figures, to businessmen, to actors, and musicians. We chose to use musicians and actors due to the importance of their relevance, and because after a bit of surface level exploratory analysis on the data, we deemed the data suitable for modelling.

Eight Steps of Forecasting

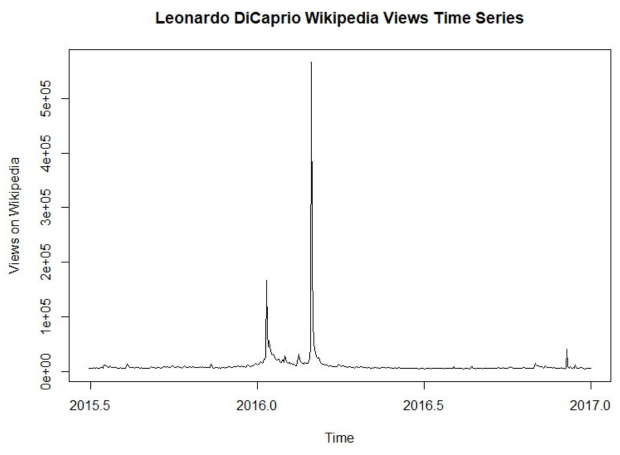
Define Goal

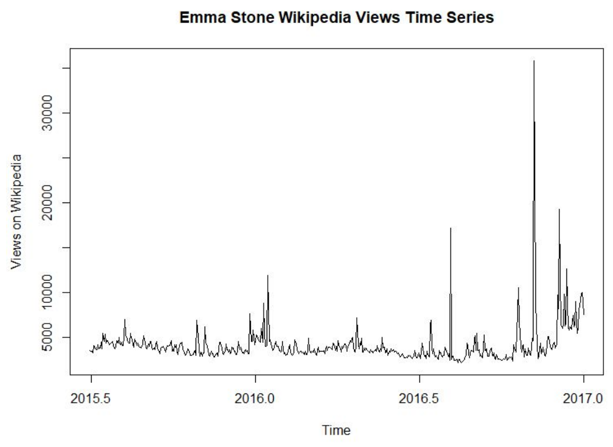
This project’s goal is to forecast the web traffic of a number of public figures, which is determined by the daily views of these figures’ Wikipedia articles over a range of a few years. These forecasts will show how the popularity of these figures changes over time. In order to do so, a model that properly accounts for factors such as trend and seasonality will be employed, assuming such factors play a significant role in a public figure’s popularity. Whichever model performs the best will then be used to perform the aforementioned forecasting.

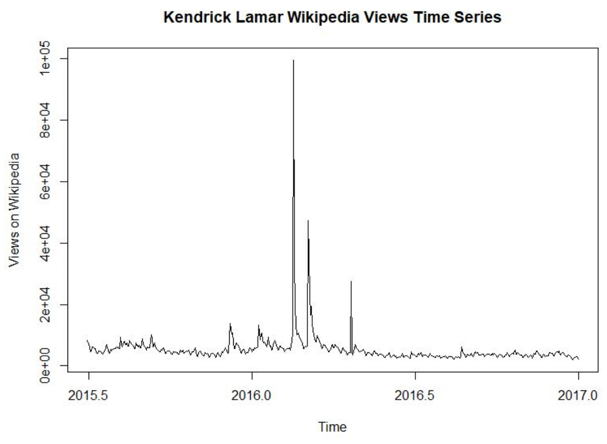
Get Data

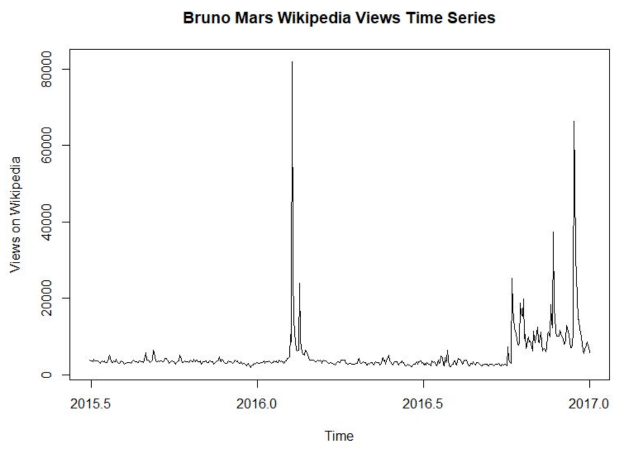
The data is obtained from Kaggle, where the competition was to use the provided data to predict the view count of a specific date range. Our model will not be designed to predict the same specific date range in particular, but will utilize the data all the same. The dataset records daily traffic from a date range of July 2015 to the end of September 2017. For the analysis, the date range of July 2015 to December 2016 will be used to create the model, and the remaining data will stand as a reference point for the model’s forecasting. The public figures in particular that this report will analyze are two actors, Emma Stone and Leonardo DiCaprio, as well as two music artists, Kendrick Lamar and Bruno Mars.

Explore and Visualize Series

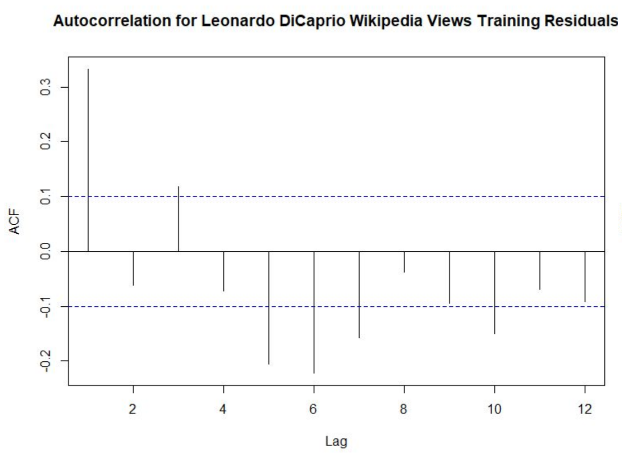


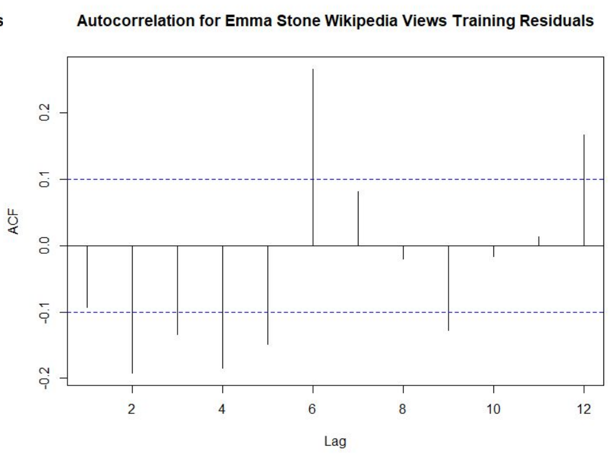


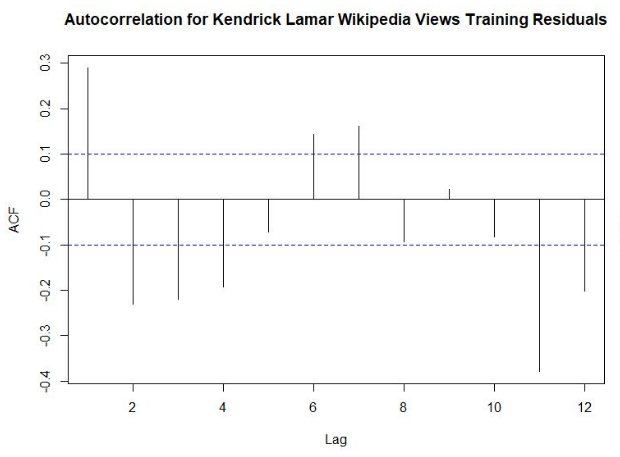


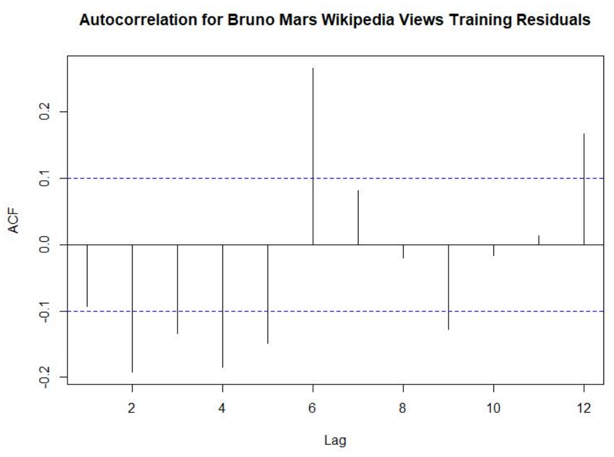


The above four plots show the data of the traffic that the four people of interest have received. For each of them there is a significant spike in traffic for at least one moment, multiple in the case of Emma Stone and Bruno Mars. These spikes in traffic most likely coincide with an event that occurred which is relevant to the person, such as the release of an album or movie. Aside from the spikes, there seems to be a stable traffic amount for the most part.









The plots above display the autocorrelation of each of the datasets. For each of them, it seems that there is at least one significant lag, and a handful near the line, showing that there *may* be reason to conclude that the data has more than just a level component. In addition, some of the autocorrelations are observed to be negative. Many of the lags are shown to be statistically significant showing evidence of autocorrelation. Because we see that much of the data is correlated to each other we can conclude that the data is predictable and not random. Further examination can be done when comparing the results of the models.

Data Preprocessing and Partitioning

The original dataset contained information for many different public figures. This dataset was thoroughly trimmed down to contain only people whose Wikipedia traffic can be used in a more business-oriented way. In addition, other elements that were not relevant to the report were removed, such as the type of web browser, because this part of the data did not affect the analysis. For the people of interest, the excess data was trimmed down to just their Wikipedia page traffic, which was recorded on a daily view basis. Each of the datasets have 550 entries, one for each day, from a date range of July 2015 to the end of September 2017.

The data was then partitioned into a training set with a size of 385 entries, and the remaining 165 entries were the validation set.

Apply Forecasting and Comparing Forecasts

Seasonal Naïve

Person	**RMSE**	**MAPE**
Emma Stone	3523.847	38.256
Leonardo DiCaprio	3568.537	36.146
Kendrick Lamar	2540.538	61.674
Bruno Mars	8575.075	46.168

The seasonal naive model is a model which makes the assumption that the forecasted value of a given point is what it was one cycle ago. Because of this, it is not a very strong model and only serves as a baseline for other models.

Moving Average

K=2

Person	RMSE	MAPE
Emma Stone	530.968	7.472
Leonardo DiCaprio	3147.596	0.646
Kendrick Lamar	1393.413	41.307
Bruno Mars	8847.243	48.016

K=6

Person	RMSE	MAPE
Emma Stone	179.539	2.558
Leonardo DiCaprio	3172.239	13.292
Kendrick Lamar	624.958	14.492
Bruno Mars	8951.360	43.544

K=12

Person	RMSE	MAPE
Emma Stone	89.337	1.380
Leonardo DiCaprio	3249.088	14.566
Kendrick Lamar	624.221	14.962
Bruno Mars	9195.337	44.525

The moving average model is a model which uses the moving average at varying windows to forecast future values. In this case, windows of 2, 6, and 12 were used. In almost every case, the RMSE values were close to the seasonal naive model, showing that the moving average models are probably not the best to use to forecast, despite the lower MAPE values.

Linear Trend

Person	**RMSE**	**MAPE**
Emma Stone	857.465	13.606
Leonardo DiCaprio	31670.26	70.721
Kendrick Lamar	6059.452	36.179
Bruno Mars	4434.258	21.59

A basic forecast to attempt is a regression model which accounts for linear trends. Given the shape of the data, it could be expected that this model may not perform the best compared to others but can provide a benchmark for other, more complicated models to outperform. This is because the data shows more than just linear factors, so failing to account for the other factors will make the forecasts inaccurate.

Seasonality

Person	**RMSE**	**MAPE**
Emma Stone	189.969	1.329
Leonardo DiCaprio	395.634	1.456
Kendrick Lamar	319.322	2.078
Bruno Mars	4400.538	24.204

A regression model with seasonality alone seems to perform noticeably better for 3 out of the 4 people. For all but Bruno Mars we observe a drastic improvement in both RMSE and MAPE, in the case of Bruno Mars we observe similar values to the linear trend model. This shows that by not accounting for seasonality the model performs worse, implying that there is seasonality within this data.

Seasonality with Linear Trend

Person	RMSE	MAPE
Emma Stone	170.823	0.987
Leonardo DiCaprio	279.552	1.153
Kendrick Lamar	39.658	0.932
Bruno Mars	82.762	0.652

By combining seasonality with linear trend, we notice another improvement across the board, and even in this case we can observe that the data for Bruno Mars has come more in line with the rest and even has a better RMSE and MAPE than the others, showing how much more effective this model is compared to the previous two.

Seasonality with Quadratic Trend

Person	RMSE	MAPE
Emma Stone	166.755	0.9
Leonardo DiCaprio	188.993	0.739

Kendrick Lamar	106.971	0.677
Bruno Mars	76.075	0.62

By taking a model similar to the one above, but instead applying quadratic trend, we notice similar results to the previous model. This one is a bit more divided, where the actors have noticeably better RMSE, but the musicians do not. This could be related to how interest in the actors is affected differently than musicians with regards to releases. Since movies have more emphasis on previews and a build-up to the release compared to music, this could have an effect on how the linear trend is reflected.

ARIMA

Person	**Model**	**RMSE test**	**MAPE test**	**RMSE valid**	**MAPE valid**
Emma Stone	(0,1,2)	767.091	10.39	3698.838	29.87
Leonardo DiCaprio	(1,0,0)	29017.373	42.981	6896.817	116.169
Kendrick Lamar	(1,0,0)	5410.751	18.616	2174.555	65.457
Bruno Mars	(1,0,1)	4040.704	12.704	8821.337	48.471

The ARIMA model is best suited for data with trend, seasonal, and level components. Given how the autocorrelation plots looked for each of these public figures, it could not be confidently assumed that the data has a strong level component. This is reflected in the poorer performance of these models compared to the previous two. For each of these models, the auto ARIMA method was used, and the automatically chosen parameters are shown above.

Two-Level

Person	RMSE	MAPE
Emma Stone	6866.535	94.531
Leonardo DiCaprio	23658.33	347.08
Kendrick Lamar	13952.9	376.075
Bruno Mars	7415.236	47.859

The two level model used combines the regression model with seasonal and quadratic trend with the ARIMA model to try and account for all factors in the forecast. In this case, the ARIMA model was performing quite poorly in comparison to the regression models with seasonality and linear/quadratic trend, so the two level model also does not show strong values for RMSE and MAPE.

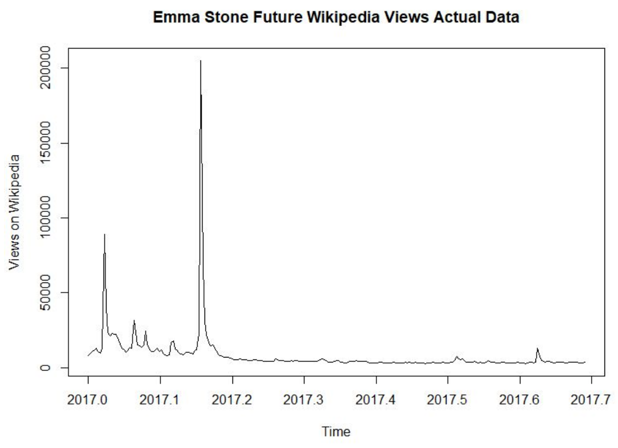
Forecasting the Data

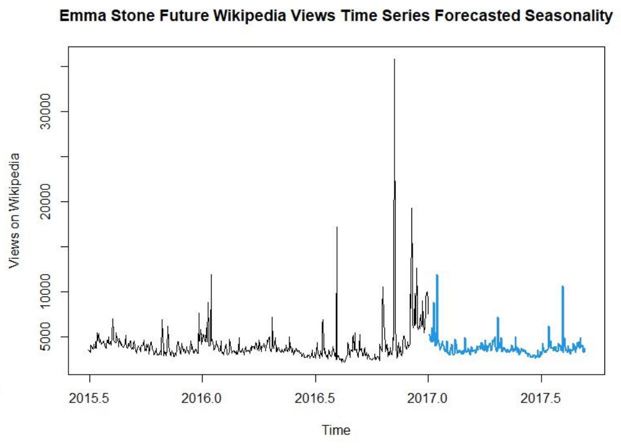
Given the performance metrics from the above models, the best models to forecast the entire dataset appear to be seasonality with linear trend, and seasonality with quadratic trend. Given this information, we will create the following forecasts: seasonal, seasonal with linear trend, and seasonal with quadratic trend. In addition, the datasets provided contained the actual data for the data we are attempting to forecast. By having access to this information, we can do a direct side by side comparison of our forecasts to the real data. By creating the models for each person, we get the following summary statistics:

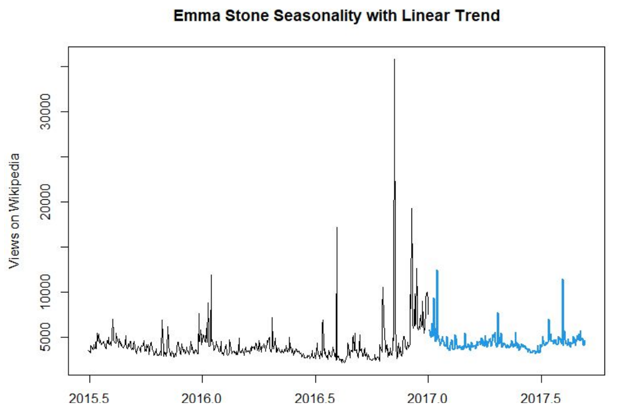
Emma Stone

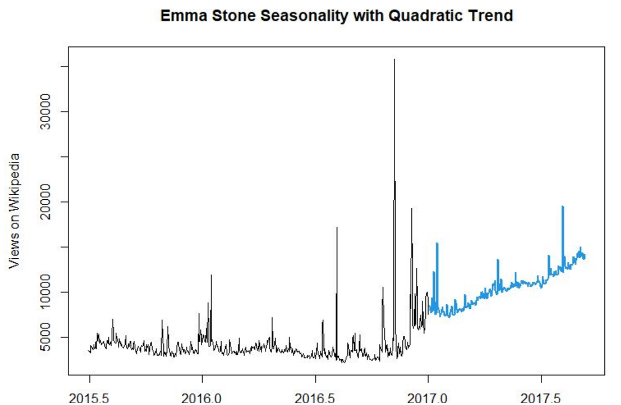
Model	p-value	Adjusted R²	RMSE	MAPE
Seasonality	1	0.5337	1827	23.253
Seasonality w/ linear trend	1	0.5452	1550.208	19.73
Seasonality w/ quadratic trend	0.8786	0.6333	231.051	2.941

Unfortunately, it seems that the forecasts for Emma stone were not very accurate, and this is reflected in the extremely poor p-values. Visualizing the forecasted traffic compared to the actual traffic, we can see that the models did not properly capture the patterns in the data.







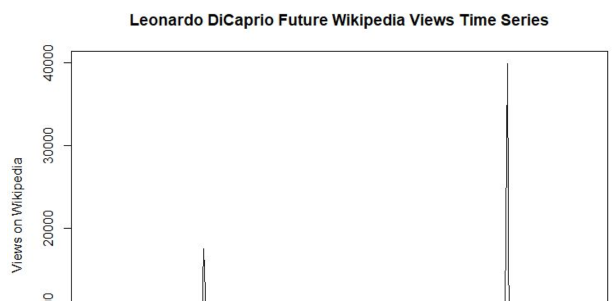


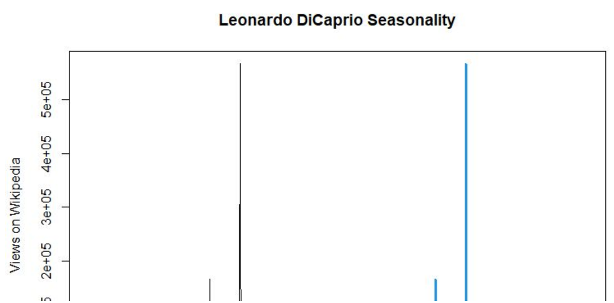
Looking at the plots above, we can see that none of the forecasts were able to predict the spike in traffic that Emma Stone received in 2017. Each of the forecasts predicted minimal traffic, with the exception of the quadratic trend forecast, which predicted a steady increase in traffic.

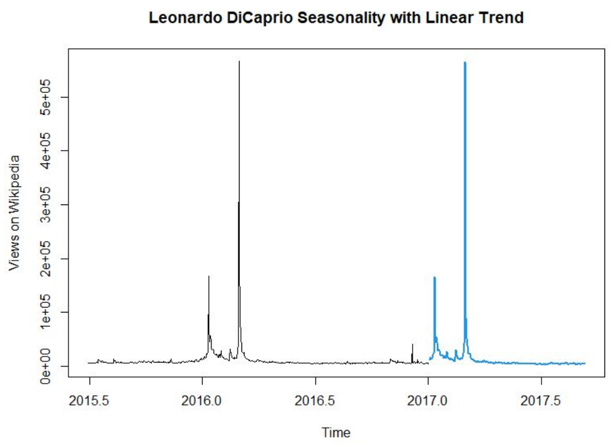
Leonardo DiCaprio

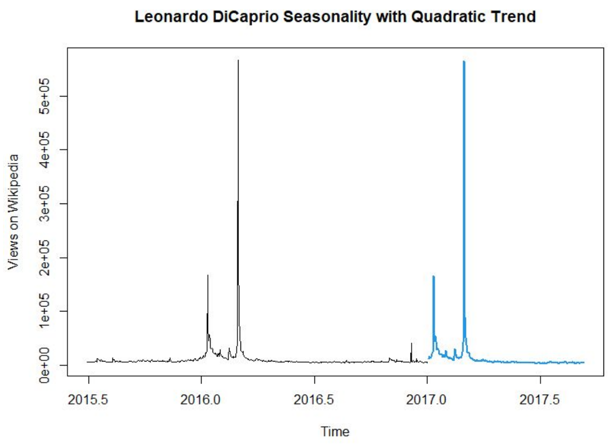
Model	**p-value**	**Adjusted R²**	**RMSE**	**MAPE**
Seasonality	<2.2e-16	0.9914	2970.5	58.822
Seasonality w/ linear trend	<2.2e-16	0.9926	2294.238	45.43
Seasonality w/ quadratic trend	<2.2e-16	0.9975	2284.595	45.42

Compared to the values from Emma Stone, the forecasts for Leonardo seem significantly better. With p-values so close to 0, and adjusted R-squared so close to 1, we should expect much better results. The adjusted R-squared implies that 99% of the variation of the page traffic can be explained by the predictors of the model.







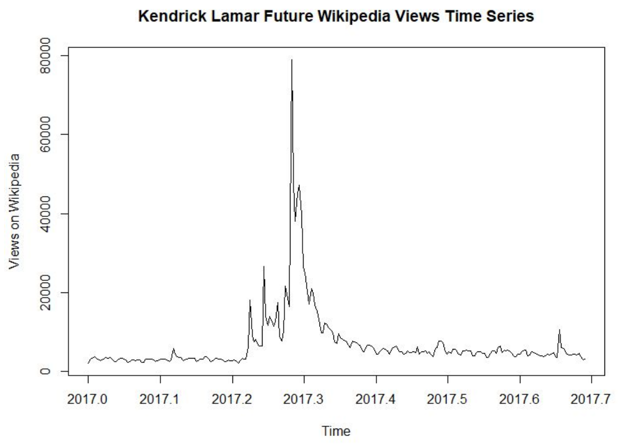


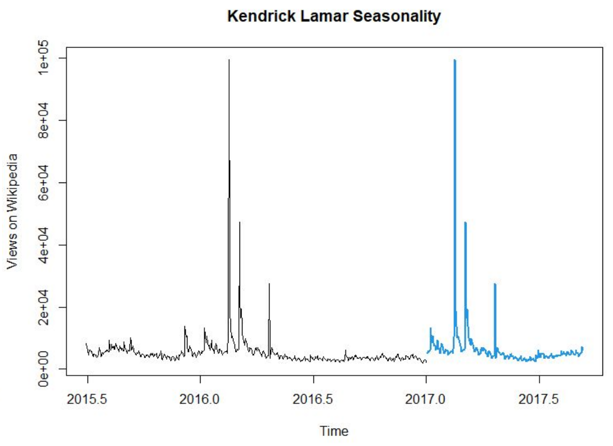
Looking at the plots above, two things are immediately noticeable. The first is how the forecasts each predicted a spike in traffic, which did indeed happen. Secondly, all the plots look very similar. This shows that trend played a very small role in each of the forecasts, if any role at all, since that was the main differing variable among each of the plots.

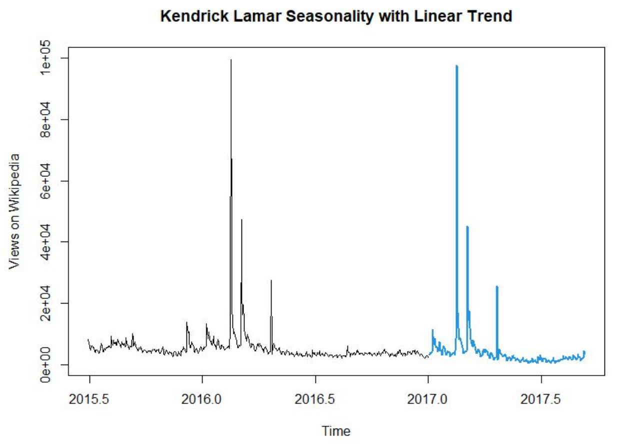
Kendrick Lamar

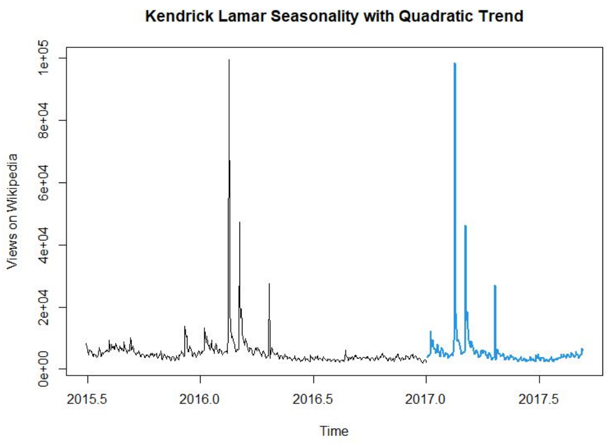
Model	**p-value**	**Adjusted R²**	**RMSE**	**MAPE**
Seasonality	<2.2e-16	.8817	1185	54.308
Seasonality w/ linear trend	<2.2e-16	.9493	222.586	10.201
Seasonality w/ quadratic trend	<2.2e-16	0.9517	548.052	25.117

Similar to Leonardo, we see p-values very close to 0, but slightly lower adjusted R-squared, in particular for Seasonality with no linear trend at all. Despite this, each of the adjusted R-squared values are very strong, even for the model with just seasonality.







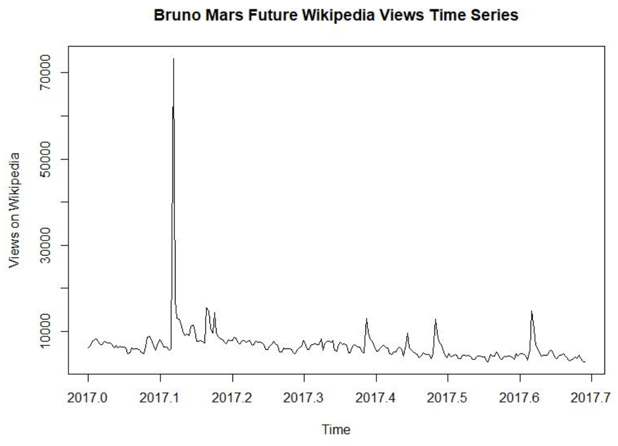


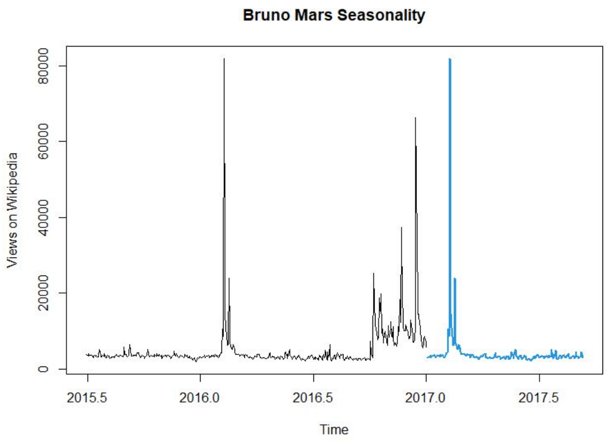
Just as observed in the summary statistics, we can see a similar outcome here compared to Leonardo. Each of the forecasts looks very similar, with each forecast predicting a spike in traffic which did occur in the real data. The spike in traffic in the real data takes a different shape than the forecasts, but the forecasts still do roughly resemble what the real data looks like.

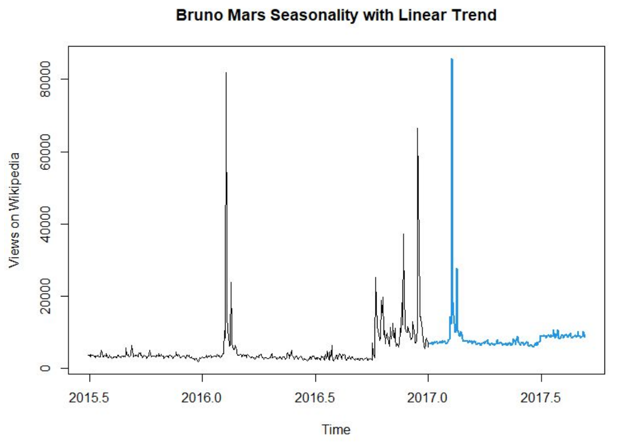
Bruno Mars

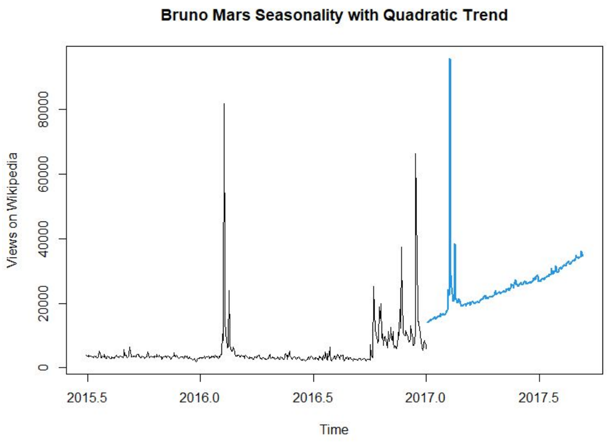
Model	p-value	Adjusted R²	RMSE	MAPE
Seasonality	.6346	-.02811	1665.5	27.112
Seasonality w/ linear trend	.01861	.1726	249.708	4.065
Seasonality w/ quadratic trend	4.891e-08	.412	3828.159	62.317

The summary statistics for Bruno Mars seems to be the most unique of them all. In both cases with linear trend we observe a strong p-value, showing statistical significance, but we observe a very weak adjusted R-squared as well. In the best case, it is .412, which is not great.









The plots above show that there was a spike forecasted in each, but the 'normal traffic', so to speak is the main difference in each. In the forecast with just seasonality, we see an overall lower amount of normal traffic compared to adding linear trend. Adding quadratic trend, we observe the forecast predicting an increase in normal traffic, which does not occur.

Implement Forecast

Best Model per Person	**Model**	**RMSE**	**MAPE**
Emma Stone	Seasonality with Quadratic Trend	166.755	0.9
Leonardo DiCaprio	Seasonality with Quadratic Trend	188.993	0.739
Kendrick Lamar	Seasonality with Linear Trend	39.658	0.932
Bruno Mars	Seasonality with Quadratic Trend	76.075	0.62

Using RMSE and MAPE as a metric for model performance, the above shows what appear to be the best. There is a common theme that the regression model with seasonality and some kind of trend, whether linear or quadratic appears to be the best.

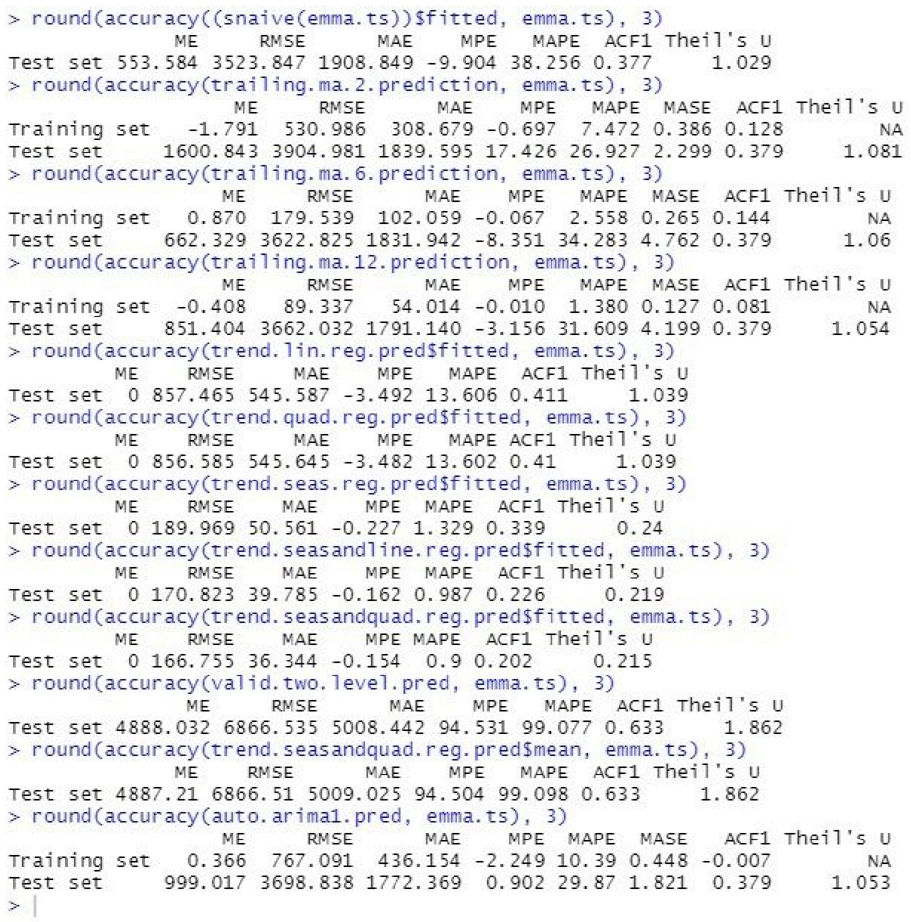
Conclusion

As stated above, a regression model with seasonality and some form of trend works best for each of the people analyzed. However, it is important to note while this paints a picture of the situation in most cases, after visualizing the data it can be seen that it is not always the case, as shown in the results given from forecasting Emma Stone. While forecasting the other three, there was some stronger resemblance to the actual data, in the case of Emma Stone, there was little resemblance at all. This shows that forecasting something such as web traffic for a handful of public figures can be tricky, given how relevant events can cause their traffic to spike seemingly randomly. Despite this, the forecasts performed have shown that to some degree it can be predicted, even if not always.

Appendix

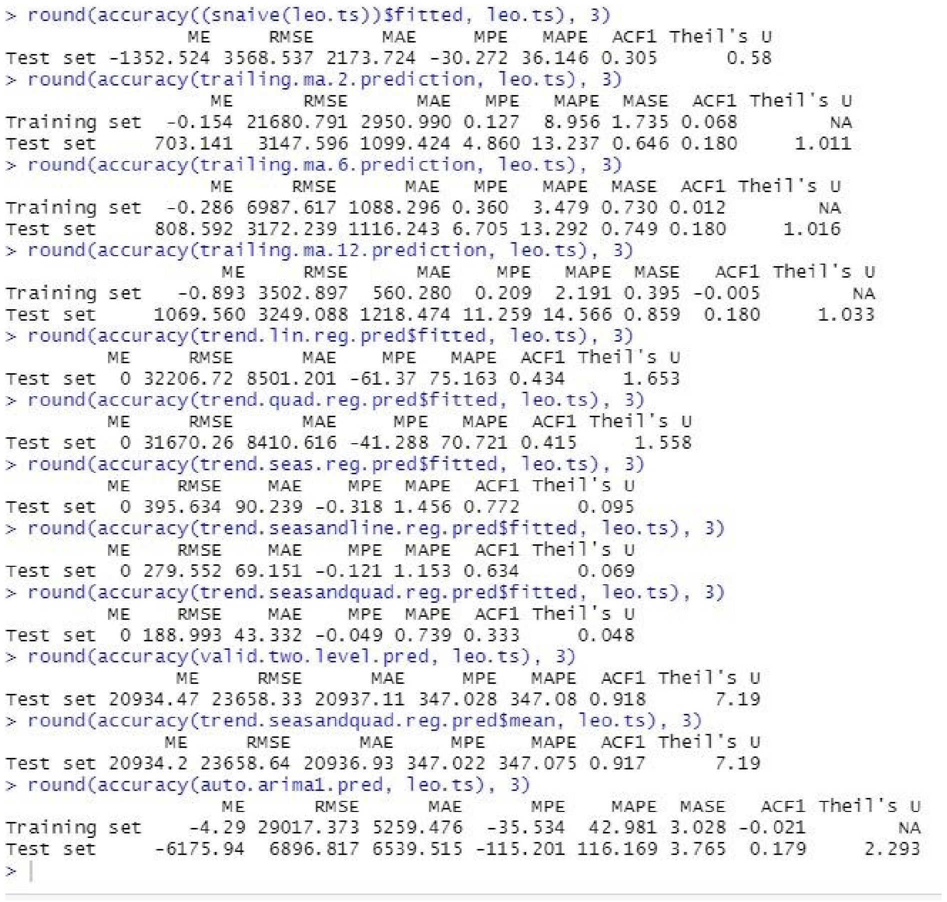
<u>Accuracy Output</u>

Emma Stone Accuracy



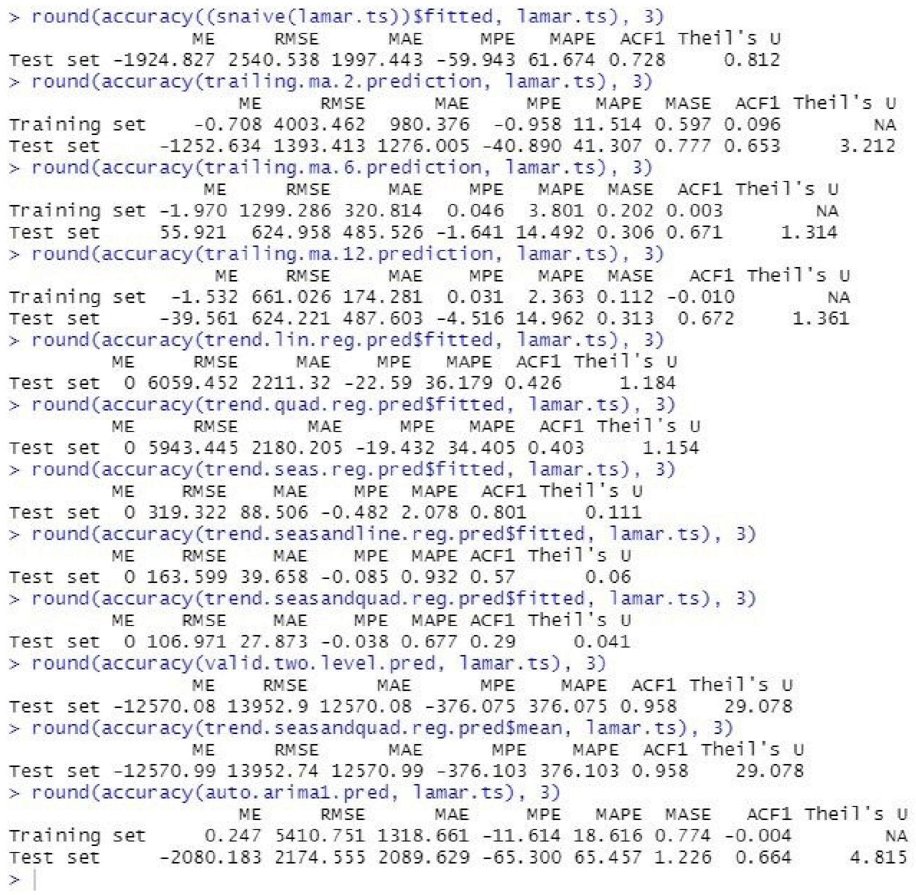
```
> round(accuracy((snaive(emma.ts))$fitted, emma.ts), 3)
              ME     RMSE      MAE    MPE   MAPE  ACF1 Theil's U
Test set 553.584 3523.847 1908.849 -9.904 38.256 0.377     1.029
> round(accuracy(trailing.ma.2.prediction, emma.ts), 3)
                   ME     RMSE      MAE    MPE   MAPE  MASE  ACF1 Theil's U
Training set   -1.791  530.986  308.679 -0.697  7.472 0.386 0.128        NA
Test set     1600.843 3904.981 1839.595 17.426 26.927 2.299 0.379     1.081
> round(accuracy(trailing.ma.6.prediction, emma.ts), 3)
                  ME     RMSE      MAE    MPE   MAPE  MASE  ACF1 Theil's U
Training set   0.870  179.539  102.059 -0.067  2.558 0.265 0.144        NA
Test set     662.329 3622.825 1831.942 -8.351 34.283 4.762 0.379      1.06
> round(accuracy(trailing.ma.12.prediction, emma.ts), 3)
                  ME     RMSE      MAE    MPE   MAPE  MASE  ACF1 Theil's U
Training set  -0.408   89.337   54.014 -0.010  1.380 0.127 0.081        NA
Test set     851.404 3662.032 1791.140 -3.156 31.609 4.199 0.379     1.054
> round(accuracy(trend.lin.reg.pred$fitted, emma.ts), 3)
         ME    RMSE     MAE    MPE   MAPE  ACF1 Theil's U
Test set  0 857.465 545.587 -3.492 13.606 0.411     1.039
> round(accuracy(trend.quad.reg.pred$fitted, emma.ts), 3)
         ME    RMSE     MAE    MPE   MAPE ACF1 Theil's U
Test set  0 856.585 545.645 -3.482 13.602 0.41     1.039
> round(accuracy(trend.seas.reg.pred$fitted, emma.ts), 3)
         ME    RMSE    MAE    MPE  MAPE  ACF1 Theil's U
Test set  0 189.969 50.561 -0.227 1.329 0.339      0.24
> round(accuracy(trend.seasandline.reg.pred$fitted, emma.ts), 3)
         ME    RMSE    MAE    MPE  MAPE  ACF1 Theil's U
Test set  0 170.823 39.785 -0.162 0.987 0.226     0.219
> round(accuracy(trend.seasandquad.reg.pred$fitted, emma.ts), 3)
         ME    RMSE    MAE    MPE MAPE  ACF1 Theil's U
Test set  0 166.755 36.344 -0.154  0.9 0.202     0.215
> round(accuracy(valid.two.level.pred, emma.ts), 3)
               ME     RMSE      MAE    MPE   MAPE  ACF1 Theil's U
Test set 4888.032 6866.535 5008.442 94.531 99.077 0.633     1.862
> round(accuracy(trend.seasandquad.reg.pred$mean, emma.ts), 3)
              ME    RMSE      MAE    MPE   MAPE  ACF1 Theil's U
Test set 4887.21 6866.51 5009.025 94.504 99.098 0.633     1.862
> round(accuracy(auto.arima1.pred, emma.ts), 3)
                  ME     RMSE      MAE    MPE  MAPE  MASE   ACF1 Theil's U
Training set   0.366  767.091  436.154 -2.249 10.39 0.448 -0.007        NA
Test set     999.017 3698.838 1772.369  0.902 29.87 1.821  0.379     1.053
> |
```

Leonardo DiCaprio Accuracy



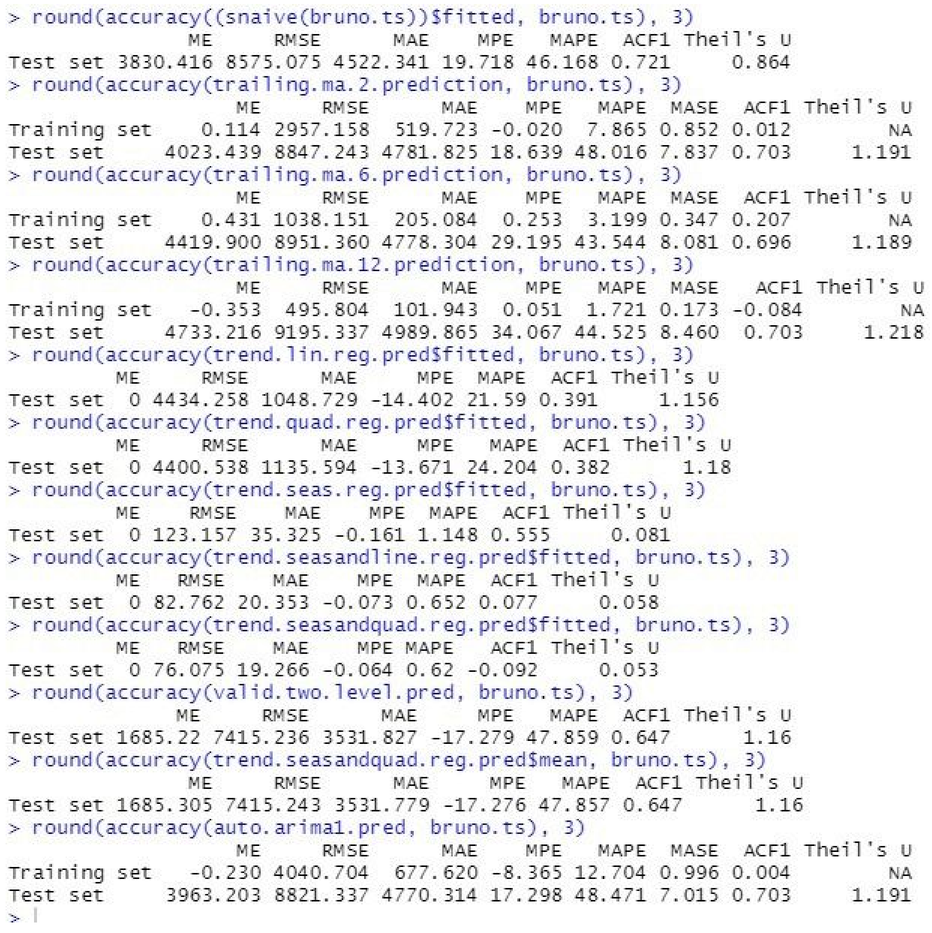
```
> round(accuracy((snaive(leo.ts))$fitted, leo.ts), 3)
                ME     RMSE      MAE     MPE   MAPE  ACF1 Theil's U
Test set -1352.524 3568.537 2173.724 -30.272 36.146 0.305      0.58
> round(accuracy(trailing.ma.2.prediction, leo.ts), 3)
                 ME      RMSE      MAE   MPE   MAPE  MASE  ACF1 Theil's U
Training set  -0.154 21680.791 2950.990 0.127  8.956 1.735 0.068        NA
Test set     703.141  3147.596 1099.424 4.860 13.237 0.646 0.180     1.011
> round(accuracy(trailing.ma.6.prediction, leo.ts), 3)
                 ME     RMSE      MAE   MPE   MAPE  MASE  ACF1 Theil's U
Training set  -0.286 6987.617 1088.296 0.360  3.479 0.730 0.012        NA
Test set     808.592 3172.239 1116.243 6.705 13.292 0.749 0.180     1.016
> round(accuracy(trailing.ma.12.prediction, leo.ts), 3)
                  ME     RMSE      MAE    MPE   MAPE  MASE   ACF1 Theil's U
Training set   -0.893 3502.897  560.280  0.209  2.191 0.395 -0.005        NA
Test set     1069.560 3249.088 1218.474 11.259 14.566 0.859  0.180     1.033
> round(accuracy(trend.lin.reg.pred$fitted, leo.ts), 3)
         ME     RMSE      MAE    MPE   MAPE  ACF1 Theil's U
Test set  0 32206.72 8501.201 -61.37 75.163 0.434     1.653
> round(accuracy(trend.quad.reg.pred$fitted, leo.ts), 3)
         ME     RMSE      MAE     MPE   MAPE  ACF1 Theil's U
Test set  0 31670.26 8410.616 -41.288 70.721 0.415     1.558
> round(accuracy(trend.seas.reg.pred$fitted, leo.ts), 3)
         ME    RMSE    MAE    MPE  MAPE  ACF1 Theil's U
Test set  0 395.634 90.239 -0.318 1.456 0.772     0.095
> round(accuracy(trend.seasandline.reg.pred$fitted, leo.ts), 3)
         ME    RMSE    MAE    MPE  MAPE  ACF1 Theil's U
Test set  0 279.552 69.151 -0.121 1.153 0.634     0.069
> round(accuracy(trend.seasandquad.reg.pred$fitted, leo.ts), 3)
         ME    RMSE    MAE    MPE  MAPE  ACF1 Theil's U
Test set  0 188.993 43.332 -0.049 0.739 0.333     0.048
> round(accuracy(valid.two.level.pred, leo.ts), 3)
               ME     RMSE      MAE     MPE   MAPE  ACF1 Theil's U
Test set 20934.47 23658.33 20937.11 347.028 347.08 0.918      7.19
> round(accuracy(trend.seasandquad.reg.pred$mean, leo.ts), 3)
              ME     RMSE      MAE     MPE    MAPE  ACF1 Theil's U
Test set 20934.2 23658.64 20936.93 347.022 347.075 0.917      7.19
> round(accuracy(auto.arima1.pred, leo.ts), 3)
                  ME      RMSE      MAE      MPE    MAPE  MASE   ACF1 Theil's U
Training set    -4.29 29017.373 5259.476  -35.534  42.981 3.028 -0.021        NA
Test set     -6175.94  6896.817 6539.515 -115.201 116.169 3.765  0.179     2.293
> |
```

Kendrick Lamar Accuracy



```
> round(accuracy((snaive(lamar.ts))$fitted, lamar.ts), 3)
                ME     RMSE      MAE     MPE   MAPE  ACF1 Theil's U
Test set -1924.827 2540.538 1997.443 -59.943 61.674 0.728     0.812
> round(accuracy(trailing.ma.2.prediction, lamar.ts), 3)
                    ME     RMSE      MAE     MPE   MAPE  MASE  ACF1 Theil's U
Training set    -0.708 4003.462  980.376  -0.958 11.514 0.597 0.096        NA
Test set     -1252.634 1393.413 1276.005 -40.890 41.307 0.777 0.653     3.212
> round(accuracy(trailing.ma.6.prediction, lamar.ts), 3)
                 ME     RMSE     MAE    MPE   MAPE  MASE  ACF1 Theil's U
Training set -1.970 1299.286 320.814  0.046  3.801 0.202 0.003        NA
Test set     55.921  624.958 485.526 -1.641 14.492 0.306 0.671     1.314
> round(accuracy(trailing.ma.12.prediction, lamar.ts), 3)
                  ME    RMSE     MAE    MPE   MAPE  MASE   ACF1 Theil's U
Training set  -1.532 661.026 174.281  0.031  2.363 0.112 -0.010        NA
Test set     -39.561 624.221 487.603 -4.516 14.962 0.313  0.672     1.361
> round(accuracy(trend.lin.reg.pred$fitted, lamar.ts), 3)
         ME     RMSE     MAE    MPE   MAPE  ACF1 Theil's U
Test set  0 6059.452 2211.32 -22.59 36.179 0.426     1.184
> round(accuracy(trend.quad.reg.pred$fitted, lamar.ts), 3)
         ME     RMSE      MAE     MPE   MAPE  ACF1 Theil's U
Test set  0 5943.445 2180.205 -19.432 34.405 0.403     1.154
> round(accuracy(trend.seas.reg.pred$fitted, lamar.ts), 3)
         ME    RMSE    MAE    MPE  MAPE  ACF1 Theil's U
Test set  0 319.322 88.506 -0.482 2.078 0.801     0.111
> round(accuracy(trend.seasandline.reg.pred$fitted, lamar.ts), 3)
         ME    RMSE    MAE    MPE  MAPE ACF1 Theil's U
Test set  0 163.599 39.658 -0.085 0.932 0.57      0.06
> round(accuracy(trend.seasandquad.reg.pred$fitted, lamar.ts), 3)
         ME    RMSE    MAE    MPE  MAPE ACF1 Theil's U
Test set  0 106.971 27.873 -0.038 0.677 0.29     0.041
> round(accuracy(valid.two.level.pred, lamar.ts), 3)
                ME    RMSE      MAE      MPE    MAPE  ACF1 Theil's U
Test set -12570.08 13952.9 12570.08 -376.075 376.075 0.958    29.078
> round(accuracy(trend.seasandquad.reg.pred$mean, lamar.ts), 3)
                ME     RMSE      MAE      MPE    MAPE  ACF1 Theil's U
Test set -12570.99 13952.74 12570.99 -376.103 376.103 0.958    29.078
> round(accuracy(auto.arima1.pred, lamar.ts), 3)
                    ME     RMSE      MAE     MPE   MAPE  MASE   ACF1 Theil's U
Training set     0.247 5410.751 1318.661 -11.614 18.616 0.774 -0.004        NA
Test set     -2080.183 2174.555 2089.629 -65.300 65.457 1.226  0.664     4.815
> |
```
]

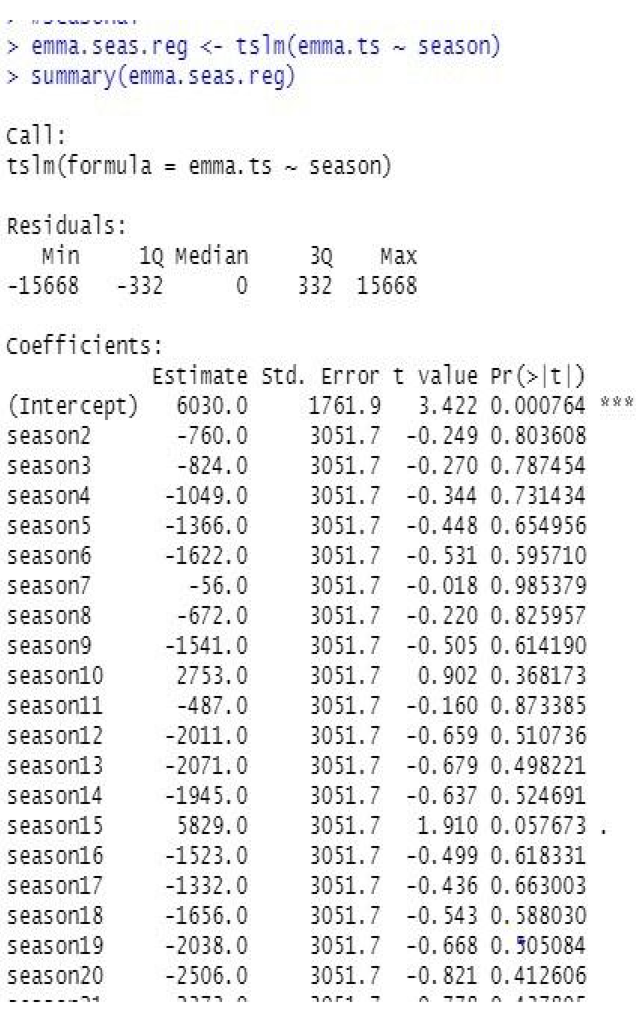
Bruno Mars Accuracy



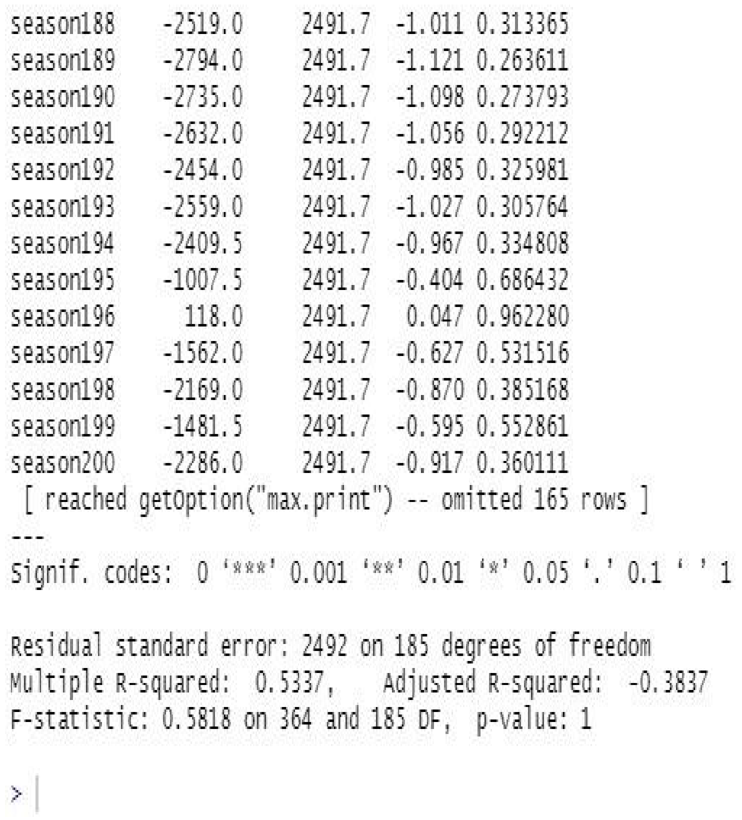
```
> round(accuracy((snaive(bruno.ts))$fitted, bruno.ts), 3)
               ME     RMSE      MAE    MPE   MAPE  ACF1 Theil's U
Test set 3830.416 8575.075 4522.341 19.718 46.168 0.721     0.864
> round(accuracy(trailing.ma.2.prediction, bruno.ts), 3)
                   ME     RMSE      MAE    MPE   MAPE  MASE  ACF1 Theil's U
Training set    0.114 2957.158  519.723 -0.020  7.865 0.852 0.012        NA
Test set     4023.439 8847.243 4781.825 18.639 48.016 7.837 0.703     1.191
> round(accuracy(trailing.ma.6.prediction, bruno.ts), 3)
                   ME     RMSE      MAE    MPE   MAPE  MASE  ACF1 Theil's U
Training set    0.431 1038.151  205.084  0.253  3.199 0.347 0.207        NA
Test set     4419.900 8951.360 4778.304 29.195 43.544 8.081 0.696     1.189
> round(accuracy(trailing.ma.12.prediction, bruno.ts), 3)
                   ME     RMSE      MAE    MPE   MAPE  MASE   ACF1 Theil's U
Training set   -0.353  495.804  101.943  0.051  1.721 0.173 -0.084        NA
Test set     4733.216 9195.337 4989.865 34.067 44.525 8.460  0.703     1.218
> round(accuracy(trend.lin.reg.pred$fitted, bruno.ts), 3)
         ME     RMSE      MAE     MPE  MAPE  ACF1 Theil's U
Test set  0 4434.258 1048.729 -14.402 21.59 0.391     1.156
> round(accuracy(trend.quad.reg.pred$fitted, bruno.ts), 3)
         ME     RMSE      MAE     MPE   MAPE  ACF1 Theil's U
Test set  0 4400.538 1135.594 -13.671 24.204 0.382      1.18
> round(accuracy(trend.seas.reg.pred$fitted, bruno.ts), 3)
         ME    RMSE    MAE    MPE  MAPE  ACF1 Theil's U
Test set  0 123.157 35.325 -0.161 1.148 0.555     0.081
> round(accuracy(trend.seasandline.reg.pred$fitted, bruno.ts), 3)
         ME   RMSE    MAE    MPE  MAPE  ACF1 Theil's U
Test set  0 82.762 20.353 -0.073 0.652 0.077     0.058
> round(accuracy(trend.seasandquad.reg.pred$fitted, bruno.ts), 3)
         ME   RMSE    MAE    MPE MAPE   ACF1 Theil's U
Test set  0 76.075 19.266 -0.064 0.62 -0.092     0.053
> round(accuracy(valid.two.level.pred, bruno.ts), 3)
              ME     RMSE      MAE     MPE   MAPE  ACF1 Theil's U
Test set 1685.22 7415.236 3531.827 -17.279 47.859 0.647      1.16
> round(accuracy(trend.seasandquad.reg.pred$mean, bruno.ts), 3)
               ME     RMSE      MAE     MPE   MAPE  ACF1 Theil's U
Test set 1685.305 7415.243 3531.779 -17.276 47.857 0.647      1.16
> round(accuracy(auto.arima1.pred, bruno.ts), 3)
                   ME     RMSE      MAE    MPE   MAPE  MASE  ACF1 Theil's U
Training set   -0.230 4040.704  677.620 -8.365 12.704 0.996 0.004        NA
Test set     3963.203 8821.337 4770.314 17.298 48.471 7.015 0.703     1.191
> |
```

Full Model Forecast Summary Statistics

Emma Stone Seasonal

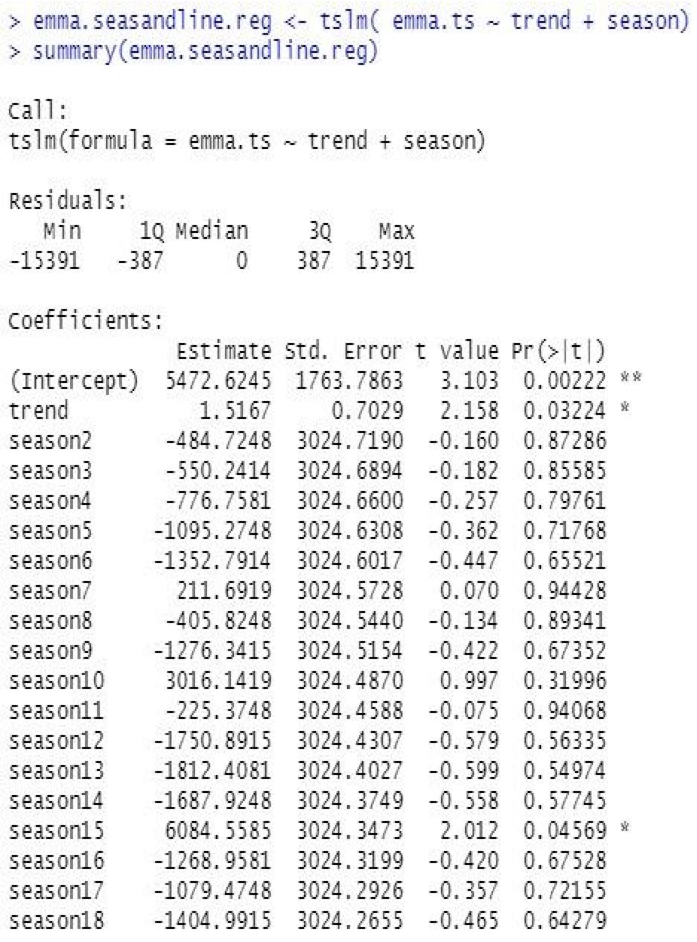


```
> emma.seas.reg <- tslm(emma.ts ~ season)
> summary(emma.seas.reg)

Call:
tslm(formula = emma.ts ~ season)

Residuals:
   Min     1Q Median     3Q    Max
-15668   -332      0    332  15668

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6030.0     1761.9   3.422 0.000764 ***
season2       -760.0     3051.7  -0.249 0.803608
season3       -824.0     3051.7  -0.270 0.787454
season4      -1049.0     3051.7  -0.344 0.731434
season5      -1366.0     3051.7  -0.448 0.654956
season6      -1622.0     3051.7  -0.531 0.595710
season7        -56.0     3051.7  -0.018 0.985379
season8       -672.0     3051.7  -0.220 0.825957
season9      -1541.0     3051.7  -0.505 0.614190
season10      2753.0     3051.7   0.902 0.368173
season11      -487.0     3051.7  -0.160 0.873385
season12     -2011.0     3051.7  -0.659 0.510736
season13     -2071.0     3051.7  -0.679 0.498221
season14     -1945.0     3051.7  -0.637 0.524691
season15      5829.0     3051.7   1.910 0.057673 .
season16     -1523.0     3051.7  -0.499 0.618331
season17     -1332.0     3051.7  -0.436 0.663003
season18     -1656.0     3051.7  -0.543 0.588030
season19     -2038.0     3051.7  -0.668 0.505084
season20     -2506.0     3051.7  -0.821 0.412606
```

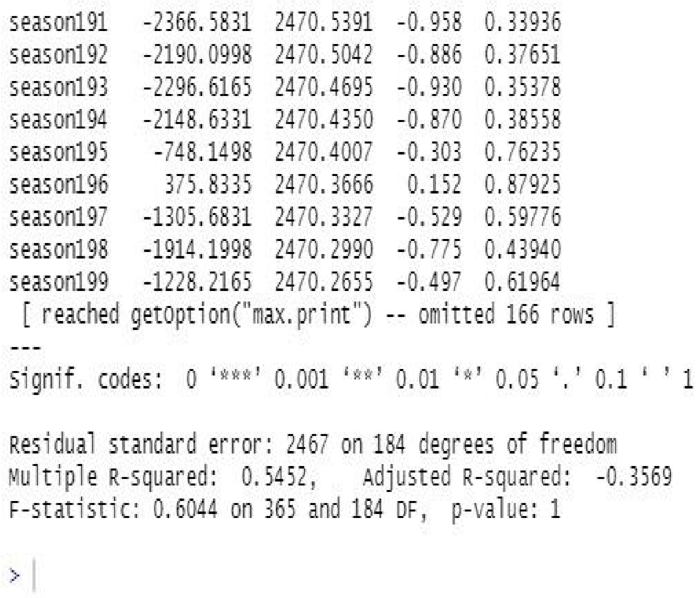


```
season188    -2519.0     2491.7  -1.011 0.313365
season189    -2794.0     2491.7  -1.121 0.263611
season190    -2735.0     2491.7  -1.098 0.273793
season191    -2632.0     2491.7  -1.056 0.292212
season192    -2454.0     2491.7  -0.985 0.325981
season193    -2559.0     2491.7  -1.027 0.305764
season194    -2409.5     2491.7  -0.967 0.334808
season195    -1007.5     2491.7  -0.404 0.686432
season196      118.0     2491.7   0.047 0.962280
season197    -1562.0     2491.7  -0.627 0.531516
season198    -2169.0     2491.7  -0.870 0.385168
season199    -1481.5     2491.7  -0.595 0.552861
season200    -2286.0     2491.7  -0.917 0.360111
 [ reached getOption("max.print") -- omitted 165 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2492 on 185 degrees of freedom
Multiple R-squared:  0.5337,	Adjusted R-squared:  -0.3837
F-statistic: 0.5818 on 364 and 185 DF,  p-value: 1

>
```

Emma Stone Seasonal with Linear Trend

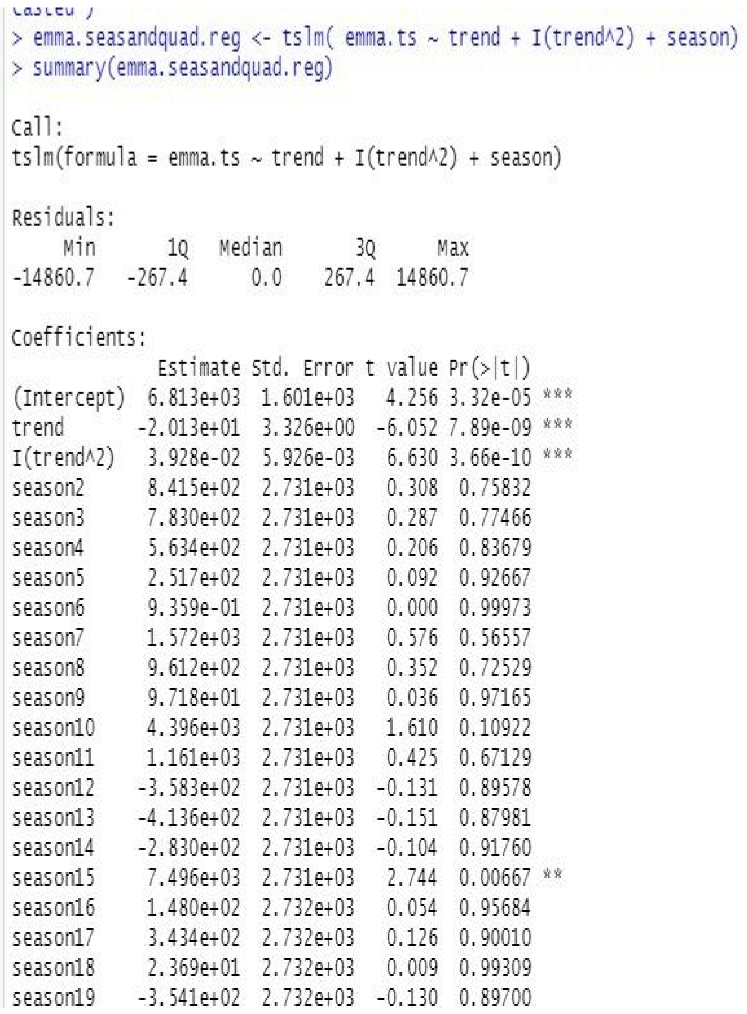


```
> emma.seasandline.reg <- tslm( emma.ts ~ trend + season)
> summary(emma.seasandline.reg)

Call:
tslm(formula = emma.ts ~ trend + season)

Residuals:
   Min     1Q Median     3Q    Max 
-15391   -387      0    387  15391 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)   
(Intercept)  5472.6245  1763.7863   3.103  0.00222 **
trend           1.5167     0.7029   2.158  0.03224 * 
season2      -484.7248  3024.7190  -0.160  0.87286   
season3      -550.2414  3024.6894  -0.182  0.85585   
season4      -776.7581  3024.6600  -0.257  0.79761   
season5     -1095.2748  3024.6308  -0.362  0.71768   
season6     -1352.7914  3024.6017  -0.447  0.65521   
season7       211.6919  3024.5728   0.070  0.94428   
season8      -405.8248  3024.5440  -0.134  0.89341   
season9     -1276.3415  3024.5154  -0.422  0.67352   
season10     3016.1419  3024.4870   0.997  0.31996   
season11     -225.3748  3024.4588  -0.075  0.94068   
season12    -1750.8915  3024.4307  -0.579  0.56335   
season13    -1812.4081  3024.4027  -0.599  0.54974   
season14    -1687.9248  3024.3749  -0.558  0.57745   
season15     6084.5585  3024.3473   2.012  0.04569 * 
season16    -1268.9581  3024.3199  -0.420  0.67528   
season17    -1079.4748  3024.2926  -0.357  0.72155   
season18    -1404.9915  3024.2655  -0.465  0.64279   
```

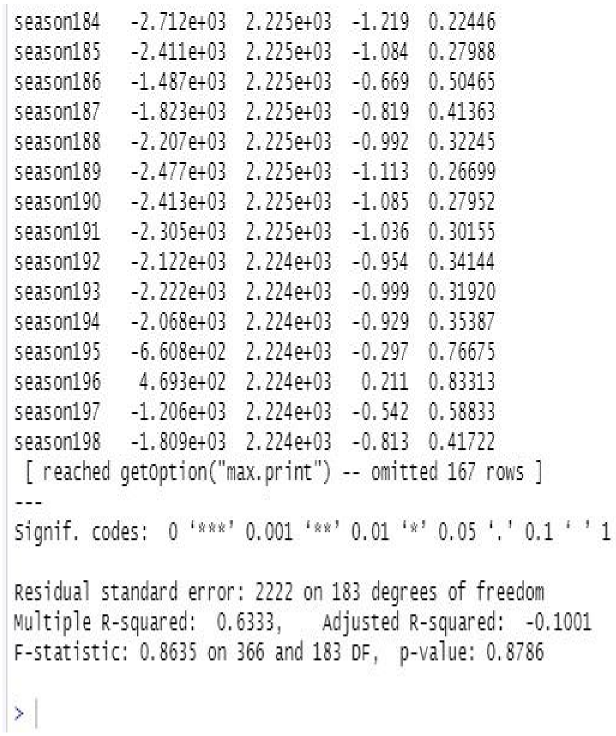


```
season191   -2366.5831  2470.5391  -0.958  0.33936   
season192   -2190.0998  2470.5042  -0.886  0.37651   
season193   -2296.6165  2470.4695  -0.930  0.35378   
season194   -2148.6331  2470.4350  -0.870  0.38558   
season195    -748.1498  2470.4007  -0.303  0.76235   
season196     375.8335  2470.3666   0.152  0.87925   
season197   -1305.6831  2470.3327  -0.529  0.59776   
season198   -1914.1998  2470.2990  -0.775  0.43940   
season199   -1228.2165  2470.2655  -0.497  0.61964   
 [ reached getOption("max.print") -- omitted 166 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2467 on 184 degrees of freedom
Multiple R-squared:  0.5452,	Adjusted R-squared:  -0.3569 
F-statistic: 0.6044 on 365 and 184 DF,  p-value: 1

> |
```

Emma Stone Seasonal with Quadratic Trend

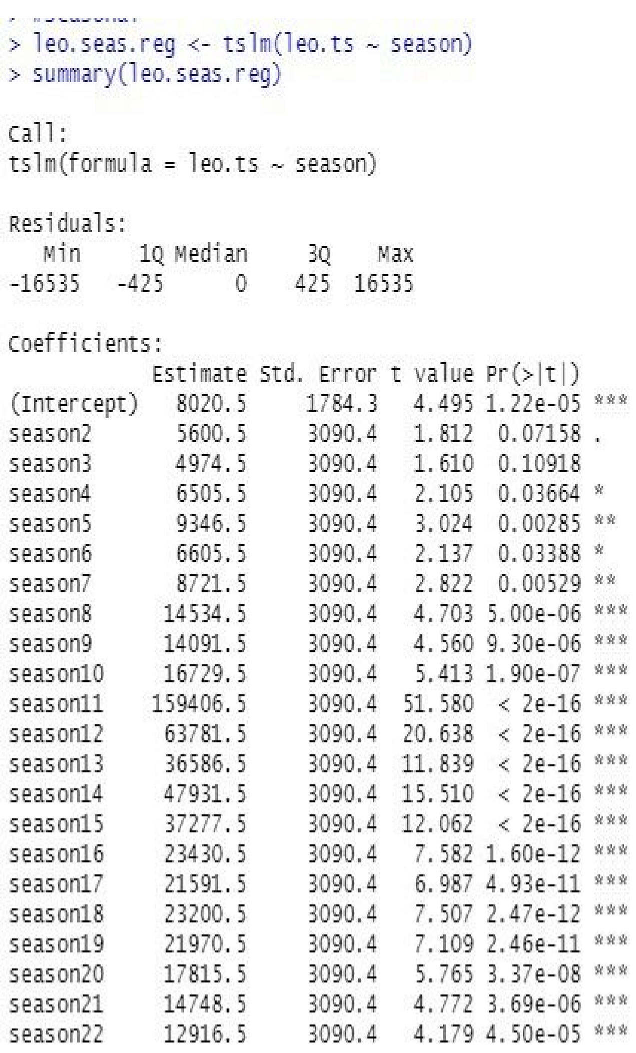


```
> emma.seasandquad.reg <- tslm( emma.ts ~ trend + I(trend^2) + season)
> summary(emma.seasandquad.reg)

Call:
tslm(formula = emma.ts ~ trend + I(trend^2) + season)

Residuals:
     Min       1Q   Median       3Q      Max 
-14860.7   -267.4      0.0    267.4  14860.7 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  6.813e+03  1.601e+03   4.256 3.32e-05 ***
trend       -2.013e+01  3.326e+00  -6.052 7.89e-09 ***
I(trend^2)   3.928e-02  5.926e-03   6.630 3.66e-10 ***
season2      8.415e+02  2.731e+03   0.308  0.75832    
season3      7.830e+02  2.731e+03   0.287  0.77466    
season4      5.634e+02  2.731e+03   0.206  0.83679    
season5      2.517e+02  2.731e+03   0.092  0.92667    
season6      9.359e-01  2.731e+03   0.000  0.99973    
season7      1.572e+03  2.731e+03   0.576  0.56557    
season8      9.612e+02  2.731e+03   0.352  0.72529    
season9      9.718e+01  2.731e+03   0.036  0.97165    
season10     4.396e+03  2.731e+03   1.610  0.10922    
season11     1.161e+03  2.731e+03   0.425  0.67129    
season12    -3.583e+02  2.731e+03  -0.131  0.89578    
season13    -4.136e+02  2.731e+03  -0.151  0.87981    
season14    -2.830e+02  2.731e+03  -0.104  0.91760    
season15     7.496e+03  2.731e+03   2.744  0.00667 ** 
season16     1.480e+02  2.732e+03   0.054  0.95684    
season17     3.434e+02  2.732e+03   0.126  0.90010    
season18     2.369e+01  2.732e+03   0.009  0.99309    
season19    -3.541e+02  2.732e+03  -0.130  0.89700    
```

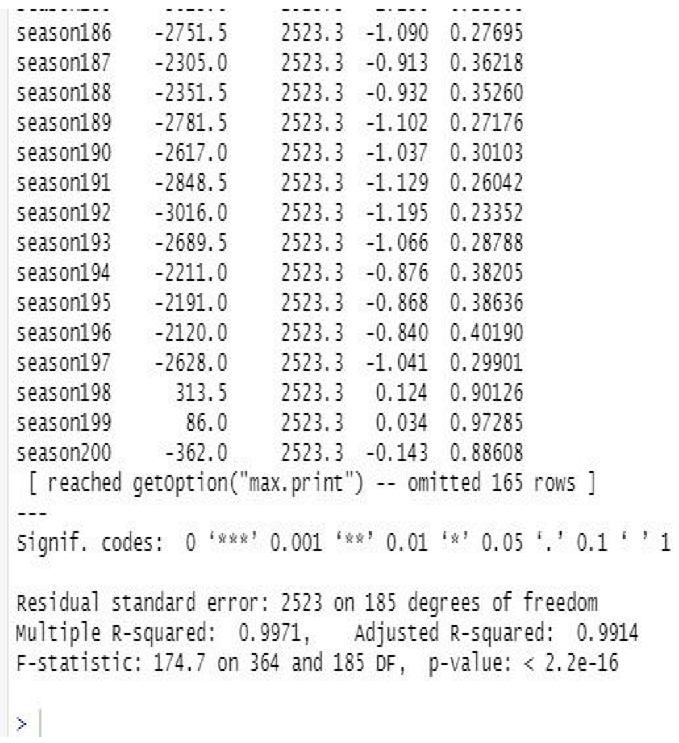


```
season184   -2.712e+03  2.225e+03  -1.219  0.22446    
season185   -2.411e+03  2.225e+03  -1.084  0.27988    
season186   -1.487e+03  2.225e+03  -0.669  0.50465    
season187   -1.823e+03  2.225e+03  -0.819  0.41363    
season188   -2.207e+03  2.225e+03  -0.992  0.32245    
season189   -2.477e+03  2.225e+03  -1.113  0.26699    
season190   -2.413e+03  2.225e+03  -1.085  0.27952    
season191   -2.305e+03  2.225e+03  -1.036  0.30155    
season192   -2.122e+03  2.224e+03  -0.954  0.34144    
season193   -2.222e+03  2.224e+03  -0.999  0.31920    
season194   -2.068e+03  2.224e+03  -0.929  0.35387    
season195   -6.608e+02  2.224e+03  -0.297  0.76675    
season196    4.693e+02  2.224e+03   0.211  0.83313    
season197   -1.206e+03  2.224e+03  -0.542  0.58833    
season198   -1.809e+03  2.224e+03  -0.813  0.41722    
 [ reached getOption("max.print") -- omitted 167 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2222 on 183 degrees of freedom
Multiple R-squared:  0.6333,	Adjusted R-squared:  -0.1001 
F-statistic: 0.8635 on 366 and 183 DF,  p-value: 0.8786

> 
```

Leonardo DiCaprio Seasonal

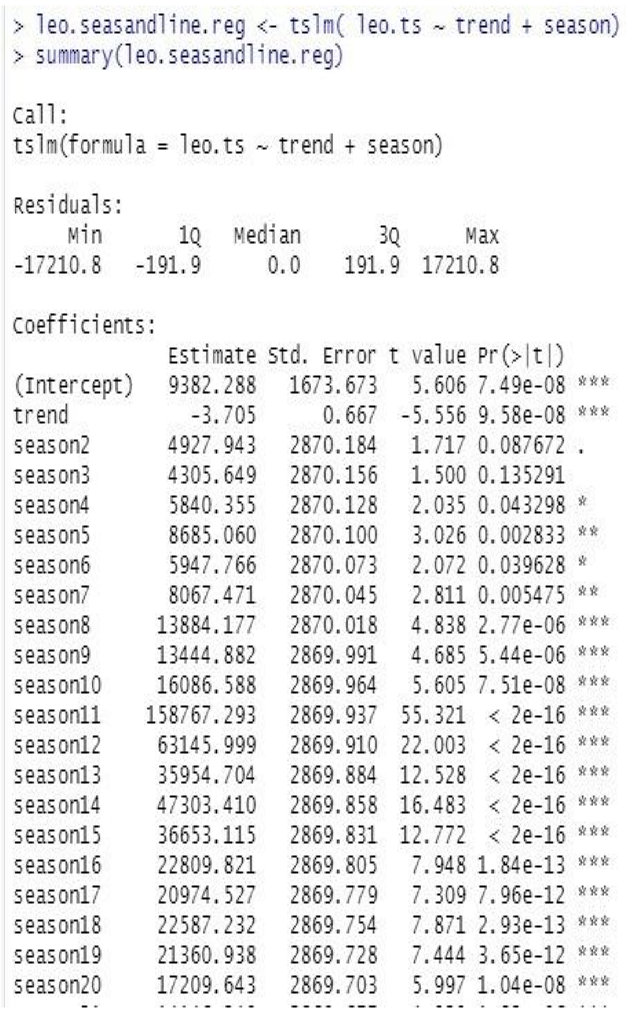


```
> leo.seas.reg <- tslm(leo.ts ~ season)
> summary(leo.seas.reg)

Call:
tslm(formula = leo.ts ~ season)

Residuals:
   Min     1Q Median     3Q    Max 
-16535   -425      0    425  16535 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   8020.5     1784.3   4.495 1.22e-05 ***
season2       5600.5     3090.4   1.812  0.07158 .  
season3       4974.5     3090.4   1.610  0.10918    
season4       6505.5     3090.4   2.105  0.03664 *  
season5       9346.5     3090.4   3.024  0.00285 ** 
season6       6605.5     3090.4   2.137  0.03388 *  
season7       8721.5     3090.4   2.822  0.00529 ** 
season8      14534.5     3090.4   4.703 5.00e-06 ***
season9      14091.5     3090.4   4.560 9.30e-06 ***
season10     16729.5     3090.4   5.413 1.90e-07 ***
season11    159406.5     3090.4  51.580  < 2e-16 ***
season12     63781.5     3090.4  20.638  < 2e-16 ***
season13     36586.5     3090.4  11.839  < 2e-16 ***
season14     47931.5     3090.4  15.510  < 2e-16 ***
season15     37277.5     3090.4  12.062  < 2e-16 ***
season16     23430.5     3090.4   7.582 1.60e-12 ***
season17     21591.5     3090.4   6.987 4.93e-11 ***
season18     23200.5     3090.4   7.507 2.47e-12 ***
season19     21970.5     3090.4   7.109 2.46e-11 ***
season20     17815.5     3090.4   5.765 3.37e-08 ***
season21     14748.5     3090.4   4.772 3.69e-06 ***
season22     12916.5     3090.4   4.179 4.50e-05 ***
```

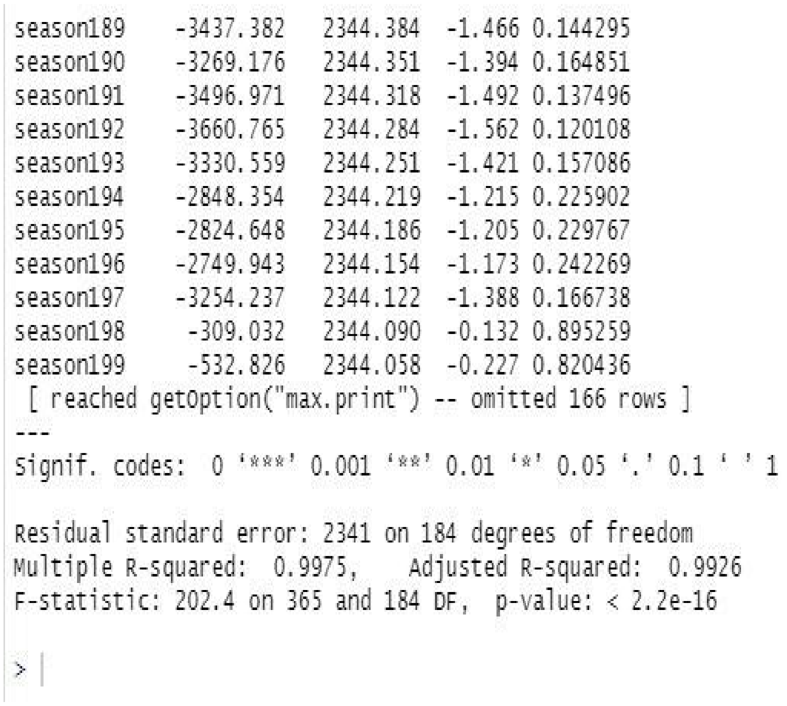


```
season186    -2751.5     2523.3  -1.090  0.27695    
season187    -2305.0     2523.3  -0.913  0.36218    
season188    -2351.5     2523.3  -0.932  0.35260    
season189    -2781.5     2523.3  -1.102  0.27176    
season190    -2617.0     2523.3  -1.037  0.30103    
season191    -2848.5     2523.3  -1.129  0.26042    
season192    -3016.0     2523.3  -1.195  0.23352    
season193    -2689.5     2523.3  -1.066  0.28788    
season194    -2211.0     2523.3  -0.876  0.38205    
season195    -2191.0     2523.3  -0.868  0.38636    
season196    -2120.0     2523.3  -0.840  0.40190    
season197    -2628.0     2523.3  -1.041  0.29901    
season198      313.5     2523.3   0.124  0.90126    
season199       86.0     2523.3   0.034  0.97285    
season200     -362.0     2523.3  -0.143  0.88608    
 [ reached getOption("max.print") -- omitted 165 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2523 on 185 degrees of freedom
Multiple R-squared:  0.9971,	Adjusted R-squared:  0.9914 
F-statistic: 174.7 on 364 and 185 DF,  p-value: < 2.2e-16

> |
```

Leonardo DiCaprio Seasonal with Linear Trend

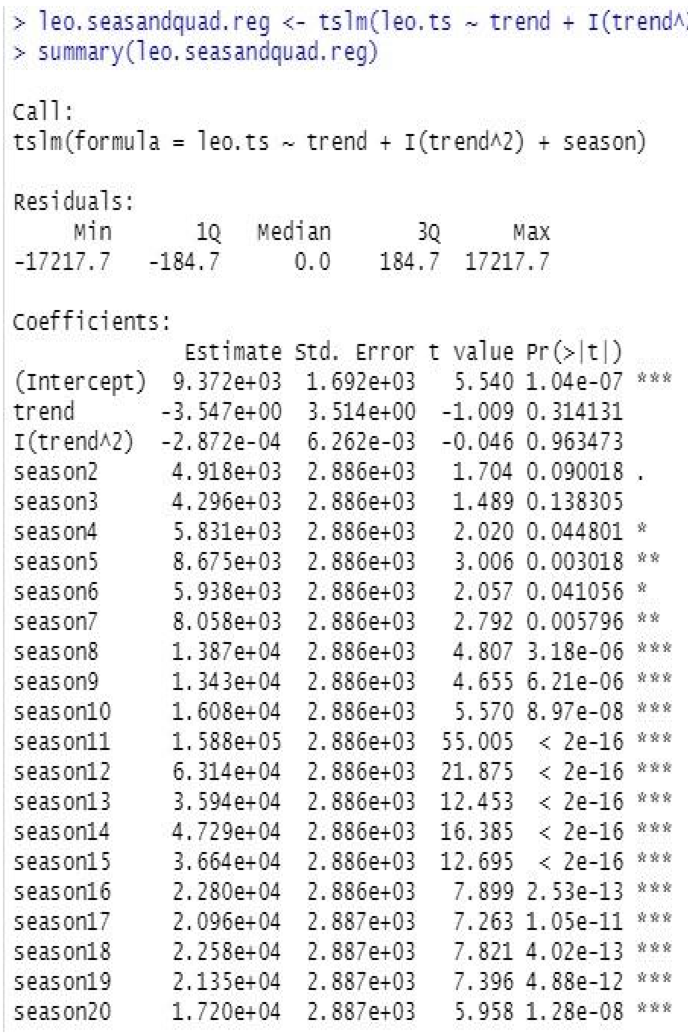


```
> leo.seasandline.reg <- tslm( leo.ts ~ trend + season)
> summary(leo.seasandline.reg)

Call:
tslm(formula = leo.ts ~ trend + season)

Residuals:
    Min      1Q  Median      3Q     Max 
-17210.8  -191.9     0.0   191.9 17210.8 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  9382.288   1673.673   5.606 7.49e-08 ***
trend          -3.705      0.667  -5.556 9.58e-08 ***
season2      4927.943   2870.184   1.717 0.087672 .  
season3      4305.649   2870.156   1.500 0.135291    
season4      5840.355   2870.128   2.035 0.043298 *  
season5      8685.060   2870.100   3.026 0.002833 ** 
season6      5947.766   2870.073   2.072 0.039628 *  
season7      8067.471   2870.045   2.811 0.005475 ** 
season8     13884.177   2870.018   4.838 2.77e-06 ***
season9     13444.882   2869.991   4.685 5.44e-06 ***
season10    16086.588   2869.964   5.605 7.51e-08 ***
season11   158767.293   2869.937  55.321  < 2e-16 ***
season12    63145.999   2869.910  22.003  < 2e-16 ***
season13    35954.704   2869.884  12.528  < 2e-16 ***
season14    47303.410   2869.858  16.483  < 2e-16 ***
season15    36653.115   2869.831  12.772  < 2e-16 ***
season16    22809.821   2869.805   7.948 1.84e-13 ***
season17    20974.527   2869.779   7.309 7.96e-12 ***
season18    22587.232   2869.754   7.871 2.93e-13 ***
season19    21360.938   2869.728   7.444 3.65e-12 ***
season20    17209.643   2869.703   5.997 1.04e-08 ***
```

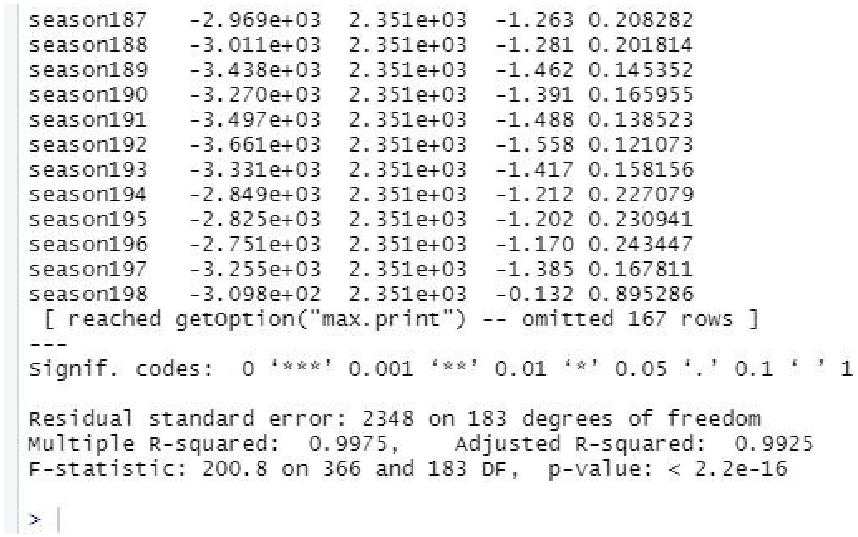


```
season189   -3437.382   2344.384  -1.466 0.144295    
season190   -3269.176   2344.351  -1.394 0.164851    
season191   -3496.971   2344.318  -1.492 0.137496    
season192   -3660.765   2344.284  -1.562 0.120108    
season193   -3330.559   2344.251  -1.421 0.157086    
season194   -2848.354   2344.219  -1.215 0.225902    
season195   -2824.648   2344.186  -1.205 0.229767    
season196   -2749.943   2344.154  -1.173 0.242269    
season197   -3254.237   2344.122  -1.388 0.166738    
season198    -309.032   2344.090  -0.132 0.895259    
season199    -532.826   2344.058  -0.227 0.820436    
 [ reached getOption("max.print") -- omitted 166 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2341 on 184 degrees of freedom
Multiple R-squared:  0.9975,	Adjusted R-squared:  0.9926 
F-statistic: 202.4 on 365 and 184 DF,  p-value: < 2.2e-16

> |
```

Leonardo DiCaprio Seasonal with Quadratic Trend

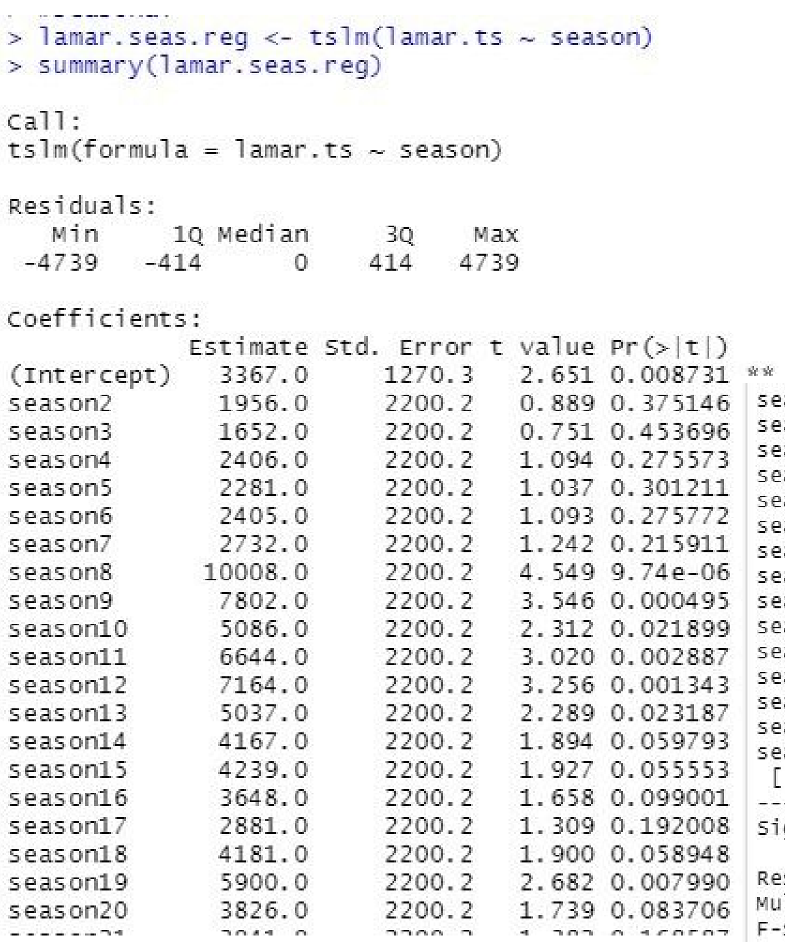


```
> leo.seasandquad.reg <- tslm(leo.ts ~ trend + I(trend^2) + season)
> summary(leo.seasandquad.reg)

Call:
tslm(formula = leo.ts ~ trend + I(trend^2) + season)

Residuals:
     Min       1Q   Median       3Q      Max
-17217.7   -184.7      0.0    184.7  17217.7

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.372e+03  1.692e+03   5.540 1.04e-07 ***
trend       -3.547e+00  3.514e+00  -1.009 0.314131
I(trend^2)  -2.872e-04  6.262e-03  -0.046 0.963473
season2      4.918e+03  2.886e+03   1.704 0.090018 .
season3      4.296e+03  2.886e+03   1.489 0.138305
season4      5.831e+03  2.886e+03   2.020 0.044801 *
season5      8.675e+03  2.886e+03   3.006 0.003018 **
season6      5.938e+03  2.886e+03   2.057 0.041056 *
season7      8.058e+03  2.886e+03   2.792 0.005796 **
season8      1.387e+04  2.886e+03   4.807 3.18e-06 ***
season9      1.343e+04  2.886e+03   4.655 6.21e-06 ***
season10     1.608e+04  2.886e+03   5.570 8.97e-08 ***
season11     1.588e+05  2.886e+03  55.005  < 2e-16 ***
season12     6.314e+04  2.886e+03  21.875  < 2e-16 ***
season13     3.594e+04  2.886e+03  12.453  < 2e-16 ***
season14     4.729e+04  2.886e+03  16.385  < 2e-16 ***
season15     3.664e+04  2.886e+03  12.695  < 2e-16 ***
season16     2.280e+04  2.886e+03   7.899 2.53e-13 ***
season17     2.096e+04  2.887e+03   7.263 1.05e-11 ***
season18     2.258e+04  2.887e+03   7.821 4.02e-13 ***
season19     2.135e+04  2.887e+03   7.396 4.88e-12 ***
season20     1.720e+04  2.887e+03   5.958 1.28e-08 ***
```

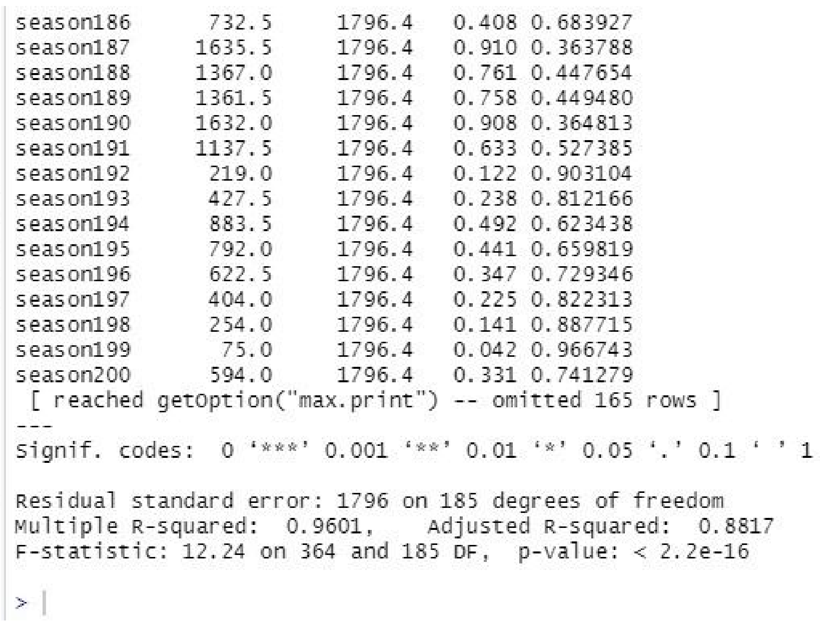


```
season187   -2.969e+03  2.351e+03  -1.263 0.208282
season188   -3.011e+03  2.351e+03  -1.281 0.201814
season189   -3.438e+03  2.351e+03  -1.462 0.145352
season190   -3.270e+03  2.351e+03  -1.391 0.165955
season191   -3.497e+03  2.351e+03  -1.488 0.138523
season192   -3.661e+03  2.351e+03  -1.558 0.121073
season193   -3.331e+03  2.351e+03  -1.417 0.158156
season194   -2.849e+03  2.351e+03  -1.212 0.227079
season195   -2.825e+03  2.351e+03  -1.202 0.230941
season196   -2.751e+03  2.351e+03  -1.170 0.243447
season197   -3.255e+03  2.351e+03  -1.385 0.167811
season198   -3.098e+02  2.351e+03  -0.132 0.895286
 [ reached getOption("max.print") -- omitted 167 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2348 on 183 degrees of freedom
Multiple R-squared:  0.9975,    Adjusted R-squared:  0.9925
F-statistic: 200.8 on 366 and 183 DF,  p-value: < 2.2e-16

> |
```

Kendrick Lamar Seasonal

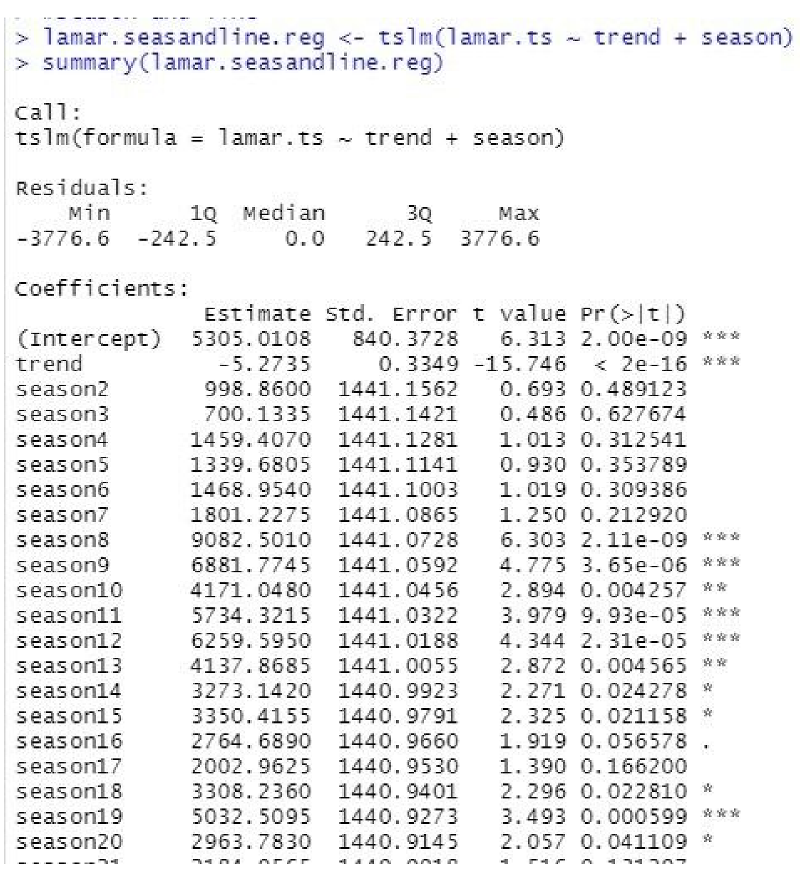


```
> lamar.seas.reg <- tslm(lamar.ts ~ season)
> summary(lamar.seas.reg)

Call:
tslm(formula = lamar.ts ~ season)

Residuals:
   Min     1Q Median     3Q    Max 
 -4739   -414      0    414   4739 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)    3367.0     1270.3   2.651 0.008731 ** 
season2        1956.0     2200.2   0.889 0.375146    
season3        1652.0     2200.2   0.751 0.453696    
season4        2406.0     2200.2   1.094 0.275573    
season5        2281.0     2200.2   1.037 0.301211    
season6        2405.0     2200.2   1.093 0.275772    
season7        2732.0     2200.2   1.242 0.215911    
season8       10008.0     2200.2   4.549 9.74e-06    
season9        7802.0     2200.2   3.546 0.000495    
season10       5086.0     2200.2   2.312 0.021899    
season11       6644.0     2200.2   3.020 0.002887    
season12       7164.0     2200.2   3.256 0.001343    
season13       5037.0     2200.2   2.289 0.023187    
season14       4167.0     2200.2   1.894 0.059793    
season15       4239.0     2200.2   1.927 0.055553    
season16       3648.0     2200.2   1.658 0.099001    
season17       2881.0     2200.2   1.309 0.192008    
season18       4181.0     2200.2   1.900 0.058948    
season19       5900.0     2200.2   2.682 0.007990    
season20       3826.0     2200.2   1.739 0.083706    
```

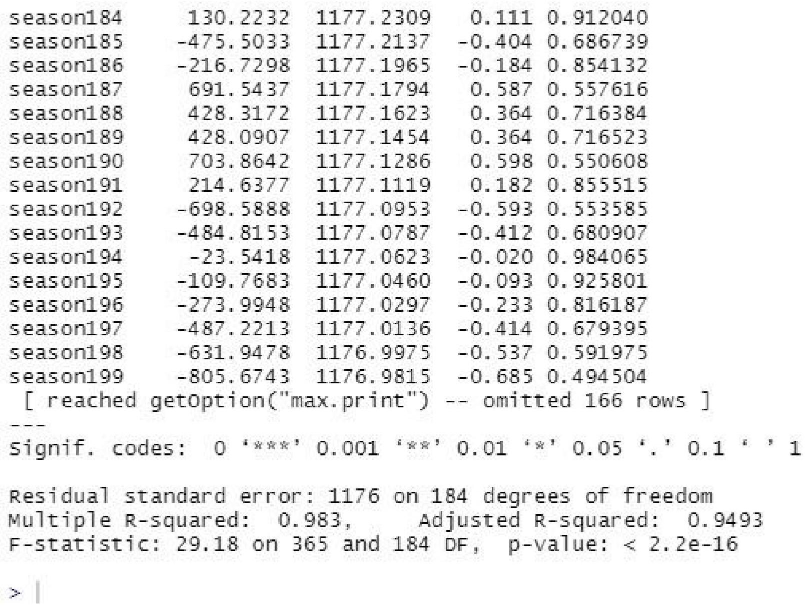


```
season186      732.5     1796.4   0.408 0.683927
season187     1635.5     1796.4   0.910 0.363788
season188     1367.0     1796.4   0.761 0.447654
season189     1361.5     1796.4   0.758 0.449480
season190     1632.0     1796.4   0.908 0.364813
season191     1137.5     1796.4   0.633 0.527385
season192      219.0     1796.4   0.122 0.903104
season193      427.5     1796.4   0.238 0.812166
season194      883.5     1796.4   0.492 0.623438
season195      792.0     1796.4   0.441 0.659819
season196      622.5     1796.4   0.347 0.729346
season197      404.0     1796.4   0.225 0.822313
season198      254.0     1796.4   0.141 0.887715
season199       75.0     1796.4   0.042 0.966743
season200      594.0     1796.4   0.331 0.741279
 [ reached getOption("max.print") -- omitted 165 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1796 on 185 degrees of freedom
Multiple R-squared:  0.9601,    Adjusted R-squared:  0.8817 
F-statistic: 12.24 on 364 and 185 DF,  p-value: < 2.2e-16

> |
```

Kendrick Lamar Seasonal with Linear Trend

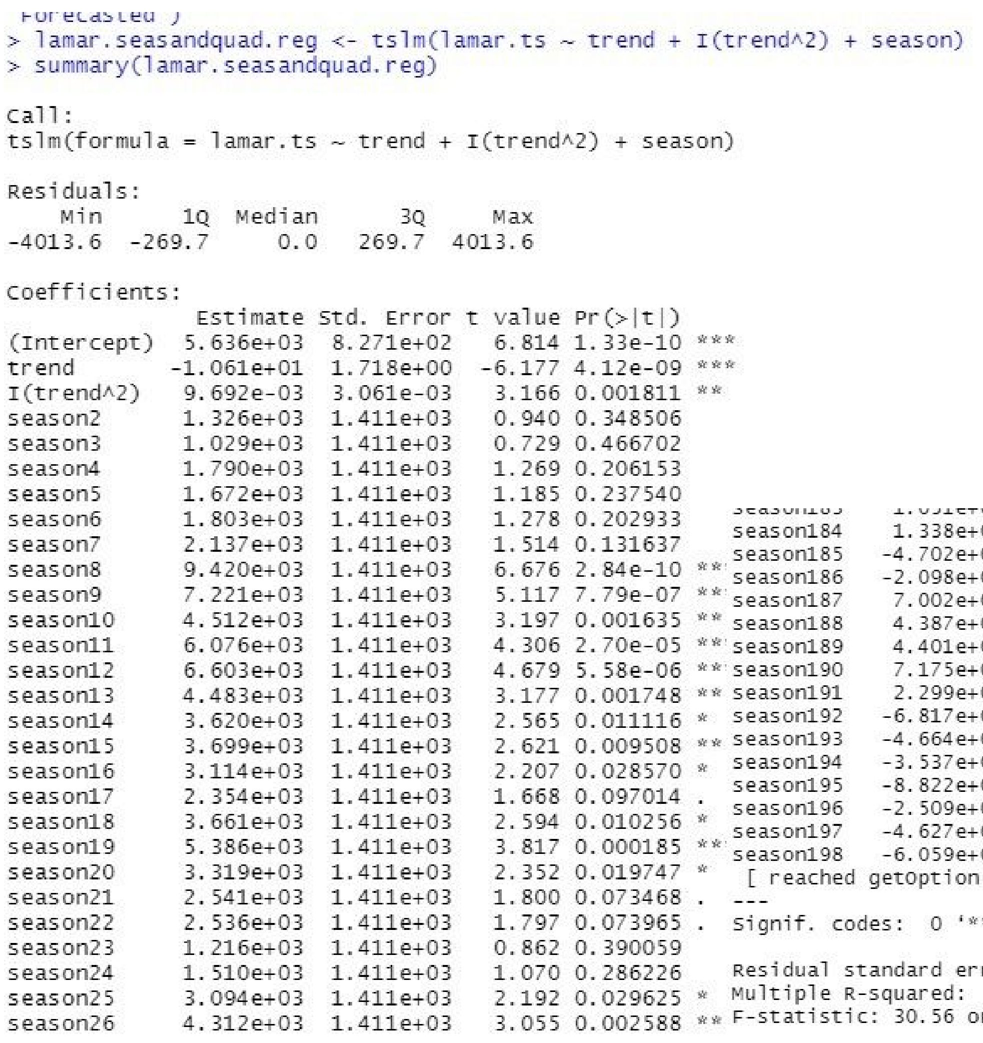


```
> lamar.seasandline.reg <- tslm(lamar.ts ~ trend + season)
> summary(lamar.seasandline.reg)

Call:
tslm(formula = lamar.ts ~ trend + season)

Residuals:
    Min      1Q  Median      3Q     Max 
-3776.6  -242.5     0.0   242.5  3776.6 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  5305.0108   840.3728   6.313 2.00e-09 ***
trend          -5.2735     0.3349 -15.746  < 2e-16 ***
season2       998.8600  1441.1562   0.693 0.489123    
season3       700.1335  1441.1421   0.486 0.627674    
season4      1459.4070  1441.1281   1.013 0.312541    
season5      1339.6805  1441.1141   0.930 0.353789    
season6      1468.9540  1441.1003   1.019 0.309386    
season7      1801.2275  1441.0865   1.250 0.212920    
season8      9082.5010  1441.0728   6.303 2.11e-09 ***
season9      6881.7745  1441.0592   4.775 3.65e-06 ***
season10     4171.0480  1441.0456   2.894 0.004257 ** 
season11     5734.3215  1441.0322   3.979 9.93e-05 ***
season12     6259.5950  1441.0188   4.344 2.31e-05 ***
season13     4137.8685  1441.0055   2.872 0.004565 ** 
season14     3273.1420  1440.9923   2.271 0.024278 *  
season15     3350.4155  1440.9791   2.325 0.021158 *  
season16     2764.6890  1440.9660   1.919 0.056578 .  
season17     2002.9625  1440.9530   1.390 0.166200    
season18     3308.2360  1440.9401   2.296 0.022810 *  
season19     5032.5095  1440.9273   3.493 0.000599 ***
season20     2963.7830  1440.9145   2.057 0.041109 *  
```

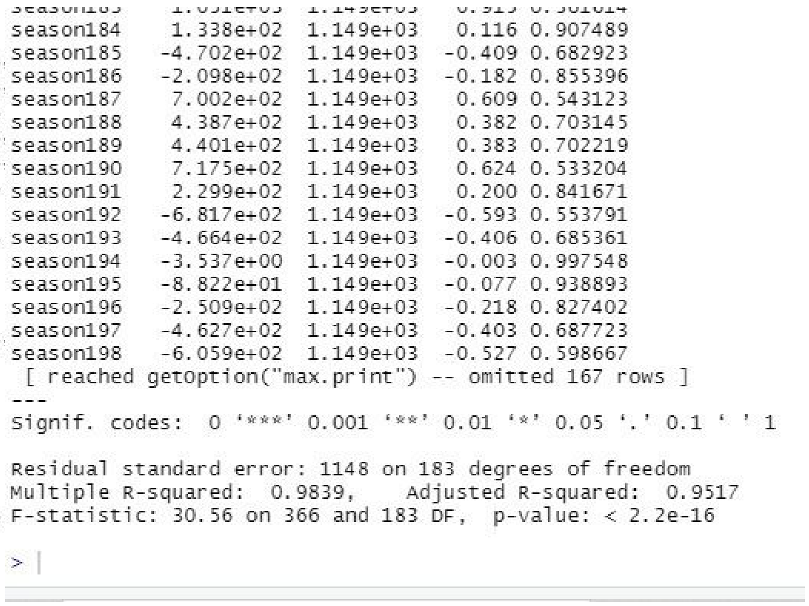


```
season184     130.2232  1177.2309   0.111 0.912040    
season185    -475.5033  1177.2137  -0.404 0.686739    
season186    -216.7298  1177.1965  -0.184 0.854132    
season187     691.5437  1177.1794   0.587 0.557616    
season188     428.3172  1177.1623   0.364 0.716384    
season189     428.0907  1177.1454   0.364 0.716523    
season190     703.8642  1177.1286   0.598 0.550608    
season191     214.6377  1177.1119   0.182 0.855515    
season192    -698.5888  1177.0953  -0.593 0.553585    
season193    -484.8153  1177.0787  -0.412 0.680907    
season194     -23.5418  1177.0623  -0.020 0.984065    
season195    -109.7683  1177.0460  -0.093 0.925801    
season196    -273.9948  1177.0297  -0.233 0.816187    
season197    -487.2213  1177.0136  -0.414 0.679395    
season198    -631.9478  1176.9975  -0.537 0.591975    
season199    -805.6743  1176.9815  -0.685 0.494504    
 [ reached getOption("max.print") -- omitted 166 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1176 on 184 degrees of freedom
Multiple R-squared:  0.983,	Adjusted R-squared:  0.9493 
F-statistic: 29.18 on 365 and 184 DF,  p-value: < 2.2e-16

> 
```

Kendrick Lamar Seasonal with Quadratic Trend

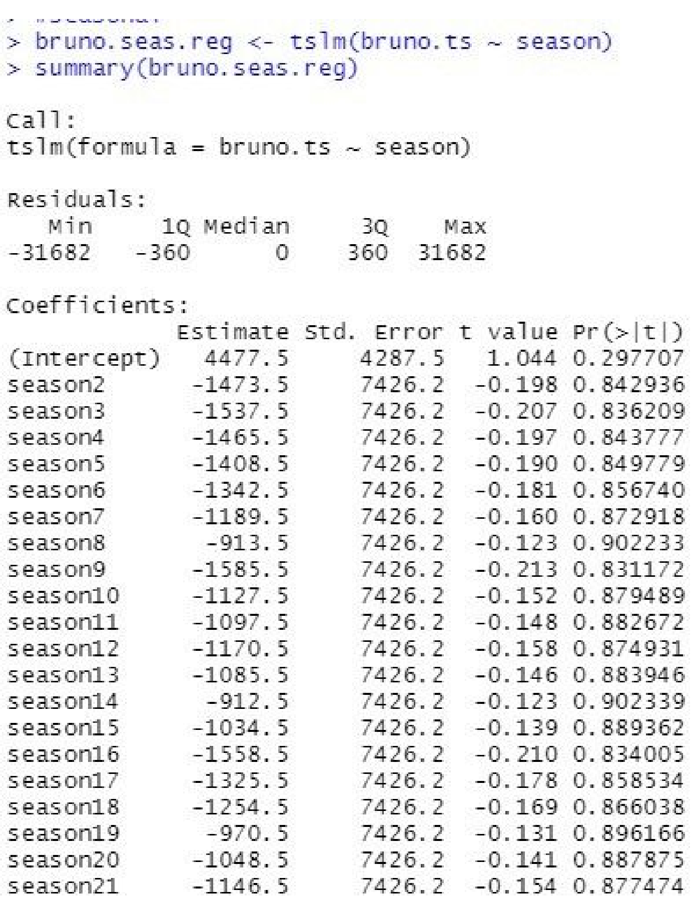


```
 Forecasted )
> lamar.seasandquad.reg <- tslm(lamar.ts ~ trend + I(trend^2) + season)
> summary(lamar.seasandquad.reg)

Call:
tslm(formula = lamar.ts ~ trend + I(trend^2) + season)

Residuals:
    Min      1Q  Median      3Q     Max 
-4013.6  -269.7     0.0   269.7  4013.6 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  5.636e+03  8.271e+02   6.814 1.33e-10 ***
trend       -1.061e+01  1.718e+00  -6.177 4.12e-09 ***
I(trend^2)   9.692e-03  3.061e-03   3.166 0.001811 ** 
season2      1.326e+03  1.411e+03   0.940 0.348506    
season3      1.029e+03  1.411e+03   0.729 0.466702    
season4      1.790e+03  1.411e+03   1.269 0.206153    
season5      1.672e+03  1.411e+03   1.185 0.237540    
season6      1.803e+03  1.411e+03   1.278 0.202933    
season7      2.137e+03  1.411e+03   1.514 0.131637    
season8      9.420e+03  1.411e+03   6.676 2.84e-10 **
season9      7.221e+03  1.411e+03   5.117 7.79e-07 **
season10     4.512e+03  1.411e+03   3.197 0.001635 **
season11     6.076e+03  1.411e+03   4.306 2.70e-05 **
season12     6.603e+03  1.411e+03   4.679 5.58e-06 **
season13     4.483e+03  1.411e+03   3.177 0.001748 **
season14     3.620e+03  1.411e+03   2.565 0.011116 * 
season15     3.699e+03  1.411e+03   2.621 0.009508 **
season16     3.114e+03  1.411e+03   2.207 0.028570 * 
season17     2.354e+03  1.411e+03   1.668 0.097014 . 
season18     3.661e+03  1.411e+03   2.594 0.010256 * 
season19     5.386e+03  1.411e+03   3.817 0.000185 **
season20     3.319e+03  1.411e+03   2.352 0.019747 * 
season21     2.541e+03  1.411e+03   1.800 0.073468 . 
season22     2.536e+03  1.411e+03   1.797 0.073965 . 
season23     1.216e+03  1.411e+03   0.862 0.390059    
season24     1.510e+03  1.411e+03   1.070 0.286226    
season25     3.094e+03  1.411e+03   2.192 0.029625 * 
season26     4.312e+03  1.411e+03   3.055 0.002588 **
```

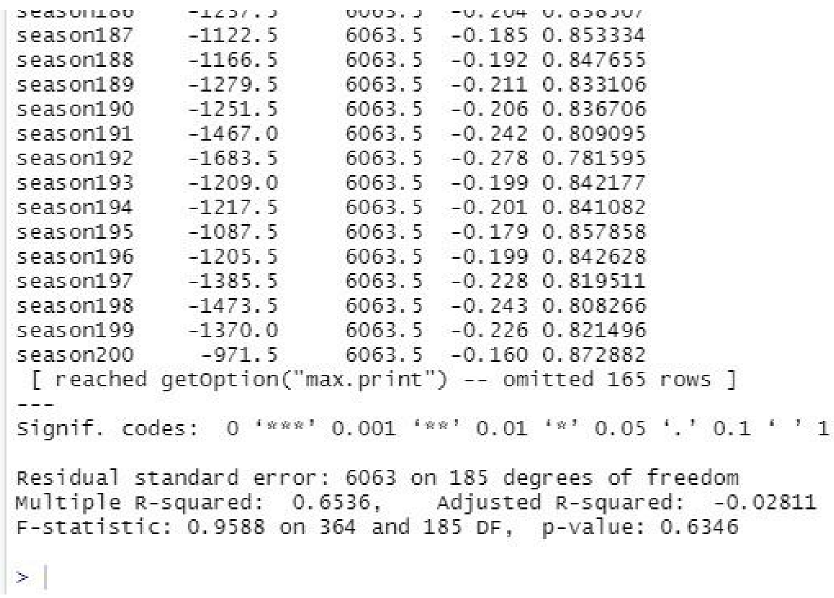


```
season184    1.338e+02  1.149e+03   0.116 0.907489
season185   -4.702e+02  1.149e+03  -0.409 0.682923
season186   -2.098e+02  1.149e+03  -0.182 0.855396
season187    7.002e+02  1.149e+03   0.609 0.543123
season188    4.387e+02  1.149e+03   0.382 0.703145
season189    4.401e+02  1.149e+03   0.383 0.702219
season190    7.175e+02  1.149e+03   0.624 0.533204
season191    2.299e+02  1.149e+03   0.200 0.841671
season192   -6.817e+02  1.149e+03  -0.593 0.553791
season193   -4.664e+02  1.149e+03  -0.406 0.685361
season194   -3.537e+00  1.149e+03  -0.003 0.997548
season195   -8.822e+01  1.149e+03  -0.077 0.938893
season196   -2.509e+02  1.149e+03  -0.218 0.827402
season197   -4.627e+02  1.149e+03  -0.403 0.687723
season198   -6.059e+02  1.149e+03  -0.527 0.598667
 [ reached getOption("max.print") -- omitted 167 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1148 on 183 degrees of freedom
Multiple R-squared:  0.9839,	Adjusted R-squared:  0.9517 
F-statistic: 30.56 on 366 and 183 DF,  p-value: < 2.2e-16

> |
```

Bruno Mars Seasonal

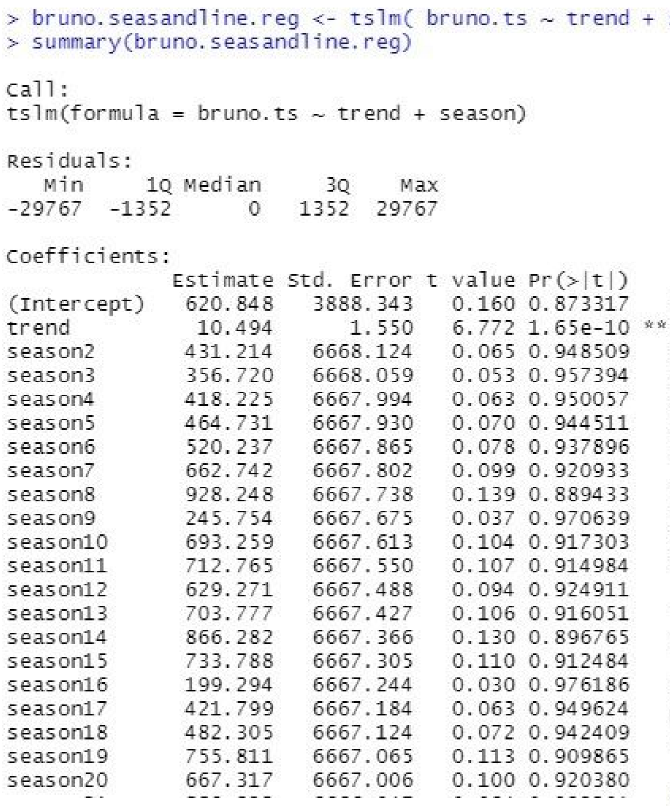


```
> bruno.seas.reg <- tslm(bruno.ts ~ season)
> summary(bruno.seas.reg)

Call:
tslm(formula = bruno.ts ~ season)

Residuals:
   Min     1Q Median     3Q    Max 
-31682   -360      0    360  31682 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4477.5     4287.5   1.044 0.297707
season2      -1473.5     7426.2  -0.198 0.842936
season3      -1537.5     7426.2  -0.207 0.836209
season4      -1465.5     7426.2  -0.197 0.843777
season5      -1408.5     7426.2  -0.190 0.849779
season6      -1342.5     7426.2  -0.181 0.856740
season7      -1189.5     7426.2  -0.160 0.872918
season8       -913.5     7426.2  -0.123 0.902233
season9      -1585.5     7426.2  -0.213 0.831172
season10     -1127.5     7426.2  -0.152 0.879489
season11     -1097.5     7426.2  -0.148 0.882672
season12     -1170.5     7426.2  -0.158 0.874931
season13     -1085.5     7426.2  -0.146 0.883946
season14      -912.5     7426.2  -0.123 0.902339
season15     -1034.5     7426.2  -0.139 0.889362
season16     -1558.5     7426.2  -0.210 0.834005
season17     -1325.5     7426.2  -0.178 0.858534
season18     -1254.5     7426.2  -0.169 0.866038
season19      -970.5     7426.2  -0.131 0.896166
season20     -1048.5     7426.2  -0.141 0.887875
season21     -1146.5     7426.2  -0.154 0.877474
```

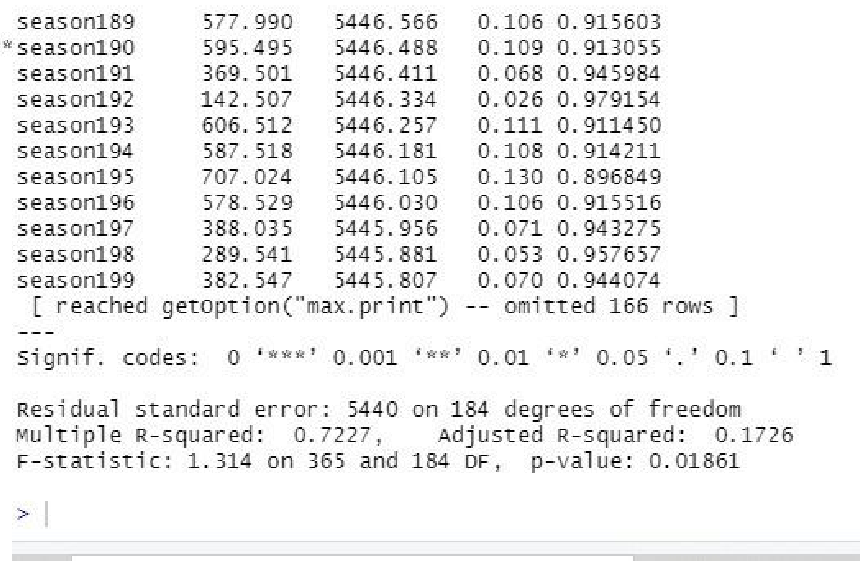


```
season186    -1237.5     6063.5  -0.204 0.838507
season187    -1122.5     6063.5  -0.185 0.853334
season188    -1166.5     6063.5  -0.192 0.847655
season189    -1279.5     6063.5  -0.211 0.833106
season190    -1251.5     6063.5  -0.206 0.836706
season191    -1467.0     6063.5  -0.242 0.809095
season192    -1683.5     6063.5  -0.278 0.781595
season193    -1209.0     6063.5  -0.199 0.842177
season194    -1217.5     6063.5  -0.201 0.841082
season195    -1087.5     6063.5  -0.179 0.857858
season196    -1205.5     6063.5  -0.199 0.842628
season197    -1385.5     6063.5  -0.228 0.819511
season198    -1473.5     6063.5  -0.243 0.808266
season199    -1370.0     6063.5  -0.226 0.821496
season200     -971.5     6063.5  -0.160 0.872882
 [ reached getOption("max.print") -- omitted 165 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6063 on 185 degrees of freedom
Multiple R-squared:  0.6536,	Adjusted R-squared:  -0.02811 
F-statistic: 0.9588 on 364 and 185 DF,  p-value: 0.6346

> 
```

Bruno Mars Seasonal with Linear Trend

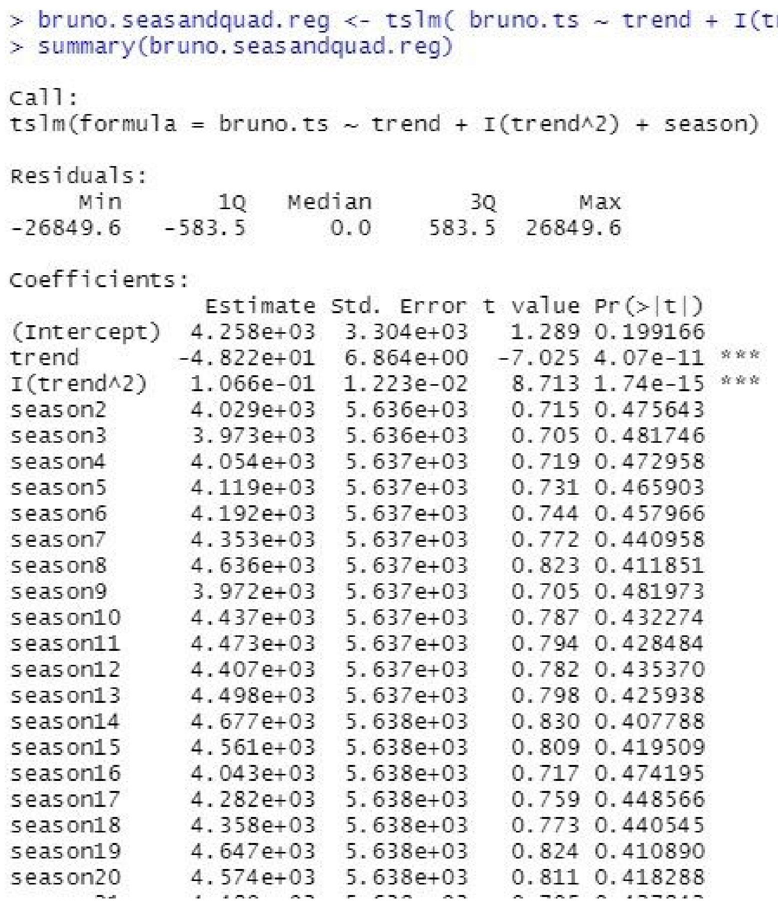


```
> bruno.seasandline.reg <- tslm( bruno.ts ~ trend + season)
> summary(bruno.seasandline.reg)

Call:
tslm(formula = bruno.ts ~ trend + season)

Residuals:
   Min     1Q Median     3Q    Max 
-29767  -1352      0   1352  29767 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)   620.848   3888.343   0.160 0.873317    
trend          10.494      1.550   6.772 1.65e-10 **
season2       431.214   6668.124   0.065 0.948509    
season3       356.720   6668.059   0.053 0.957394    
season4       418.225   6667.994   0.063 0.950057    
season5       464.731   6667.930   0.070 0.944511    
season6       520.237   6667.865   0.078 0.937896    
season7       662.742   6667.802   0.099 0.920933    
season8       928.248   6667.738   0.139 0.889433    
season9       245.754   6667.675   0.037 0.970639    
season10      693.259   6667.613   0.104 0.917303    
season11      712.765   6667.550   0.107 0.914984    
season12      629.271   6667.488   0.094 0.924911    
season13      703.777   6667.427   0.106 0.916051    
season14      866.282   6667.366   0.130 0.896765    
season15      733.788   6667.305   0.110 0.912484    
season16      199.294   6667.244   0.030 0.976186    
season17      421.799   6667.184   0.063 0.949624    
season18      482.305   6667.124   0.072 0.942409    
season19      755.811   6667.065   0.113 0.909865    
season20      667.317   6667.006   0.100 0.920380    
```

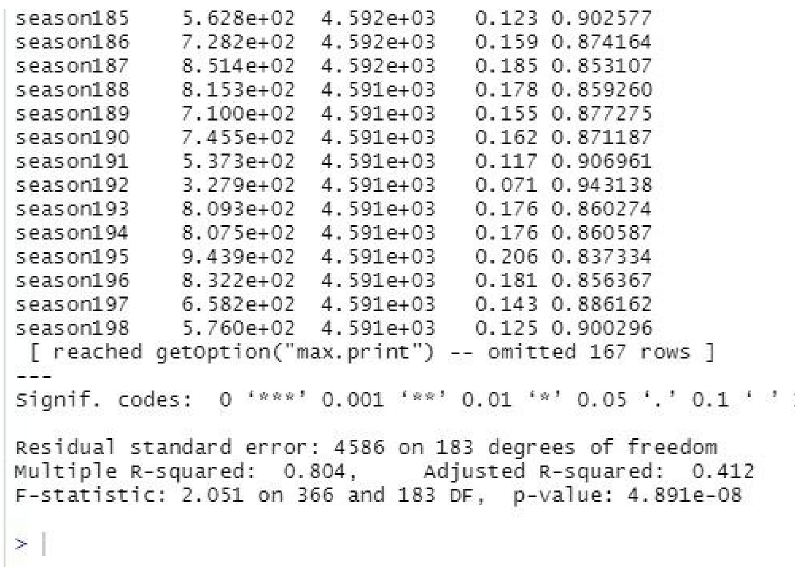


```
season189     577.990    5446.566   0.106 0.915603
season190     595.495    5446.488   0.109 0.913055
season191     369.501    5446.411   0.068 0.945984
season192     142.507    5446.334   0.026 0.979154
season193     606.512    5446.257   0.111 0.911450
season194     587.518    5446.181   0.108 0.914211
season195     707.024    5446.105   0.130 0.896849
season196     578.529    5446.030   0.106 0.915516
season197     388.035    5445.956   0.071 0.943275
season198     289.541    5445.881   0.053 0.957657
season199     382.547    5445.807   0.070 0.944074
 [ reached getOption("max.print") -- omitted 166 rows ]
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5440 on 184 degrees of freedom
Multiple R-squared:  0.7227,	Adjusted R-squared:  0.1726 
F-statistic: 1.314 on 365 and 184 DF,  p-value: 0.01861

> 
```

Bruno Mars Seasonal with Quadratic Trend

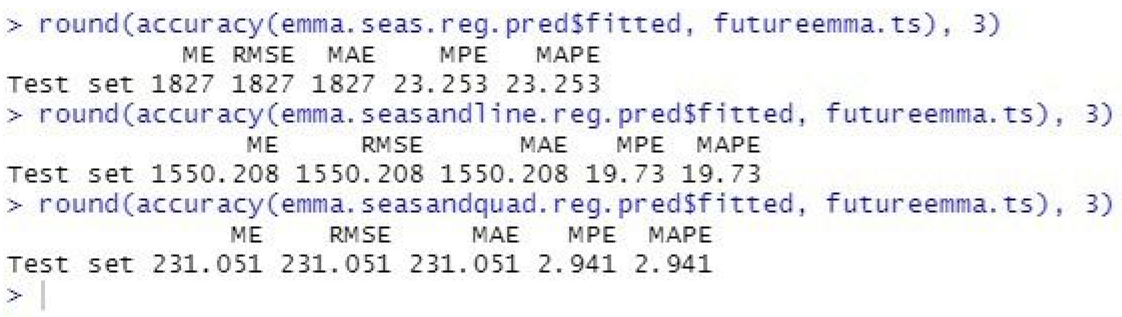


```
> bruno.seasandquad.reg <- tslm( bruno.ts ~ trend + I(trend^2) + season)
> summary(bruno.seasandquad.reg)

Call:
tslm(formula = bruno.ts ~ trend + I(trend^2) + season)

Residuals:
     Min       1Q   Median       3Q      Max 
-26849.6   -583.5      0.0    583.5  26849.6 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  4.258e+03  3.304e+03   1.289 0.199166    
trend       -4.822e+01  6.864e+00  -7.025 4.07e-11 ***
I(trend^2)   1.066e-01  1.223e-02   8.713 1.74e-15 ***
season2      4.029e+03  5.636e+03   0.715 0.475643    
season3      3.973e+03  5.636e+03   0.705 0.481746    
season4      4.054e+03  5.637e+03   0.719 0.472958    
season5      4.119e+03  5.637e+03   0.731 0.465903    
season6      4.192e+03  5.637e+03   0.744 0.457966    
season7      4.353e+03  5.637e+03   0.772 0.440958    
season8      4.636e+03  5.637e+03   0.823 0.411851    
season9      3.972e+03  5.637e+03   0.705 0.481973    
season10     4.437e+03  5.637e+03   0.787 0.432274    
season11     4.473e+03  5.637e+03   0.794 0.428484    
season12     4.407e+03  5.637e+03   0.782 0.435370    
season13     4.498e+03  5.637e+03   0.798 0.425938    
season14     4.677e+03  5.638e+03   0.830 0.407788    
season15     4.561e+03  5.638e+03   0.809 0.419509    
season16     4.043e+03  5.638e+03   0.717 0.474195    
season17     4.282e+03  5.638e+03   0.759 0.448566    
season18     4.358e+03  5.638e+03   0.773 0.440545    
season19     4.647e+03  5.638e+03   0.824 0.410890    
season20     4.574e+03  5.638e+03   0.811 0.418288    
```



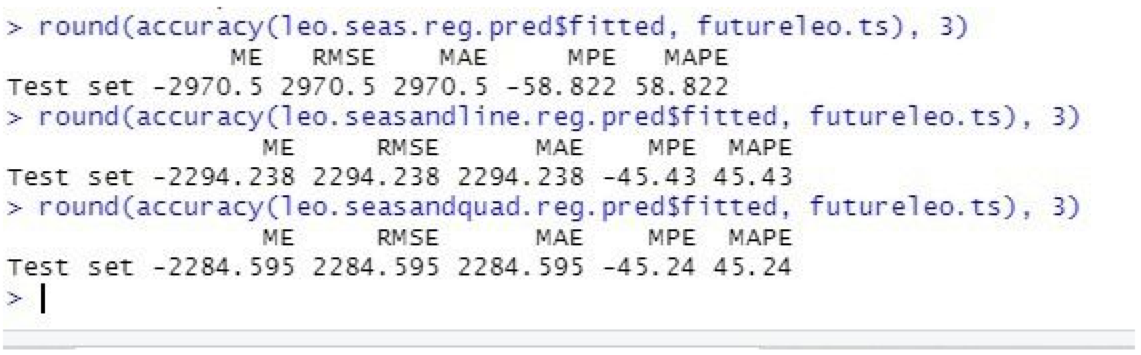
```
season185    5.628e+02  4.592e+03   0.123 0.902577    
season186    7.282e+02  4.592e+03   0.159 0.874164    
season187    8.514e+02  4.592e+03   0.185 0.853107    
season188    8.153e+02  4.591e+03   0.178 0.859260    
season189    7.100e+02  4.591e+03   0.155 0.877275    
season190    7.455e+02  4.591e+03   0.162 0.871187    
season191    5.373e+02  4.591e+03   0.117 0.906961    
season192    3.279e+02  4.591e+03   0.071 0.943138    
season193    8.093e+02  4.591e+03   0.176 0.860274    
season194    8.075e+02  4.591e+03   0.176 0.860587    
season195    9.439e+02  4.591e+03   0.206 0.837334    
season196    8.322e+02  4.591e+03   0.181 0.856367    
season197    6.582e+02  4.591e+03   0.143 0.886162    
season198    5.760e+02  4.591e+03   0.125 0.900296    
 [ reached getOption("max.print") -- omitted 167 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’

Residual standard error: 4586 on 183 degrees of freedom
Multiple R-squared:  0.804,	Adjusted R-squared:  0.412 
F-statistic: 2.051 on 366 and 183 DF,  p-value: 4.891e-08

> 
```

Emma Stone Full Data Accuracy



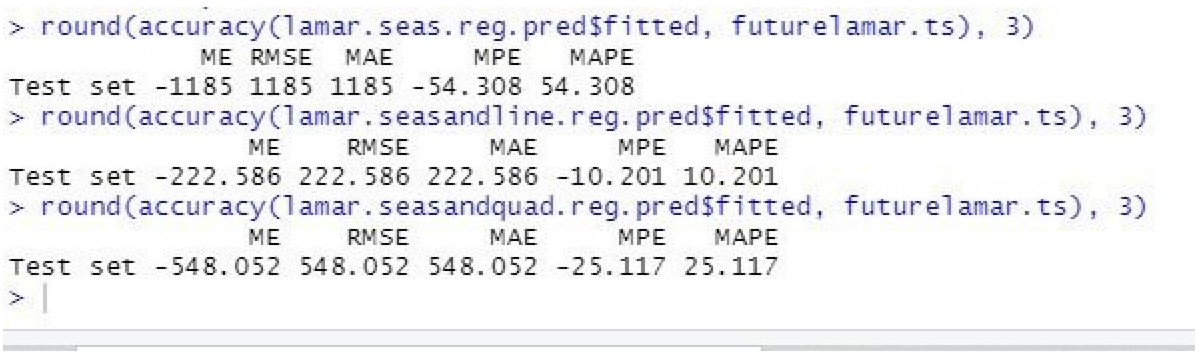
```
> round(accuracy(emma.seas.reg.pred$fitted, futureemma.ts), 3)
           ME RMSE  MAE    MPE   MAPE
Test set 1827 1827 1827 23.253 23.253
> round(accuracy(emma.seasandline.reg.pred$fitted, futureemma.ts), 3)
               ME     RMSE      MAE   MPE  MAPE
Test set 1550.208 1550.208 1550.208 19.73 19.73
> round(accuracy(emma.seasandquad.reg.pred$fitted, futureemma.ts), 3)
              ME    RMSE     MAE   MPE  MAPE
Test set 231.051 231.051 231.051 2.941 2.941
> 
```

Leonardo DiCaprio Full Data Accuracy



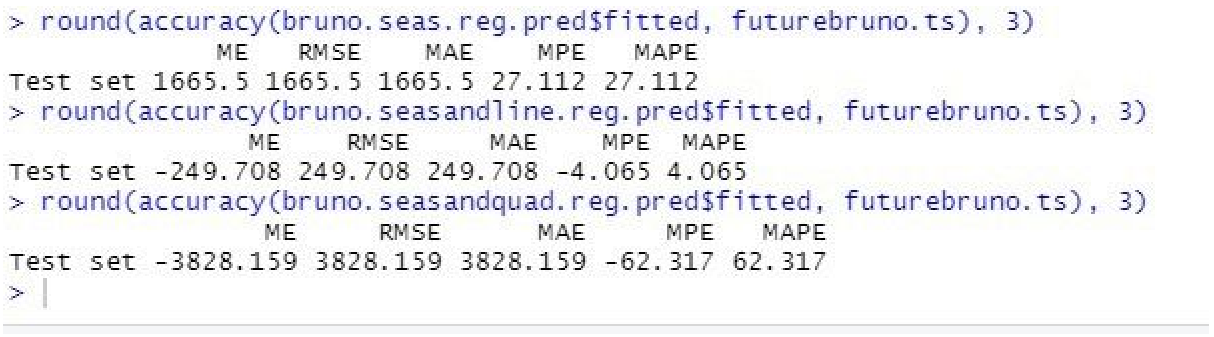
```
> round(accuracy(leo.seas.reg.pred$fitted, futureleo.ts), 3)
              ME   RMSE    MAE     MPE   MAPE
Test set -2970.5 2970.5 2970.5 -58.822 58.822
> round(accuracy(leo.seasandline.reg.pred$fitted, futureleo.ts), 3)
                ME     RMSE      MAE    MPE  MAPE
Test set -2294.238 2294.238 2294.238 -45.43 45.43
> round(accuracy(leo.seasandquad.reg.pred$fitted, futureleo.ts), 3)
                ME     RMSE      MAE    MPE  MAPE
Test set -2284.595 2284.595 2284.595 -45.24 45.24
> 
```

Kendrick Lamar Full Data Accuracy



```
> round(accuracy(lamar.seas.reg.pred$fitted, futurelamar.ts), 3)
            ME RMSE  MAE     MPE   MAPE
Test set -1185 1185 1185 -54.308 54.308
> round(accuracy(lamar.seasandline.reg.pred$fitted, futurelamar.ts), 3)
               ME    RMSE     MAE     MPE   MAPE
Test set -222.586 222.586 222.586 -10.201 10.201
> round(accuracy(lamar.seasandquad.reg.pred$fitted, futurelamar.ts), 3)
               ME    RMSE     MAE     MPE   MAPE
Test set -548.052 548.052 548.052 -25.117 25.117
> 
```

Bruno Mars Full Data Accuracy



```
> round(accuracy(bruno.seas.reg.pred$fitted, futurebruno.ts), 3)
             ME   RMSE    MAE    MPE   MAPE
Test set 1665.5 1665.5 1665.5 27.112 27.112
> round(accuracy(bruno.seasandline.reg.pred$fitted, futurebruno.ts), 3)
               ME    RMSE     MAE    MPE  MAPE
Test set -249.708 249.708 249.708 -4.065 4.065
> round(accuracy(bruno.seasandquad.reg.pred$fitted, futurebruno.ts), 3)
                ME     RMSE      MAE     MPE   MAPE
Test set -3828.159 3828.159 3828.159 -62.317 62.317
> 
```