Understanding SARIMA (More Time Series Modeling)

We Investigate How ARIMA's Big Brother, SARIMA, Improves Our Forecasting Accuracy

The future is uncertain. But there are still some things that we can be reasonably sure of — Christmas and Thanksgiving come every year, summers are filled with sunny days and vacationing families (at least they used to be), and every February forgetful boyfriends and husbands scramble for chocolates and that last available bouquet of long stemmed roses.

The seasons matter when it comes to analyzing time series. Certain events happen every year:

- Holidays
- Summer vacation
- Back to school week
- Festivals and annual conferences
- Major sporting events
- Annual bonuses

Some happen once a month:

- Rent or mortgage payments
- Credit card bills
- Our paychecks (some are bi-monthly)

Or even weekly:

- Fridays (yes something as ordinary as Fridays can be a statistically significant factor in models)
- Taco Tuesdays (just kidding)

So if we know that something is likely to happen at a regular cadence, and to impact our target variable in a similar fashion each time it occurs, we should factor that in when we build our model. In time series, this recurrence of impactful events at a constant frequency is known as **seasonality**.

ARIMA Refresher

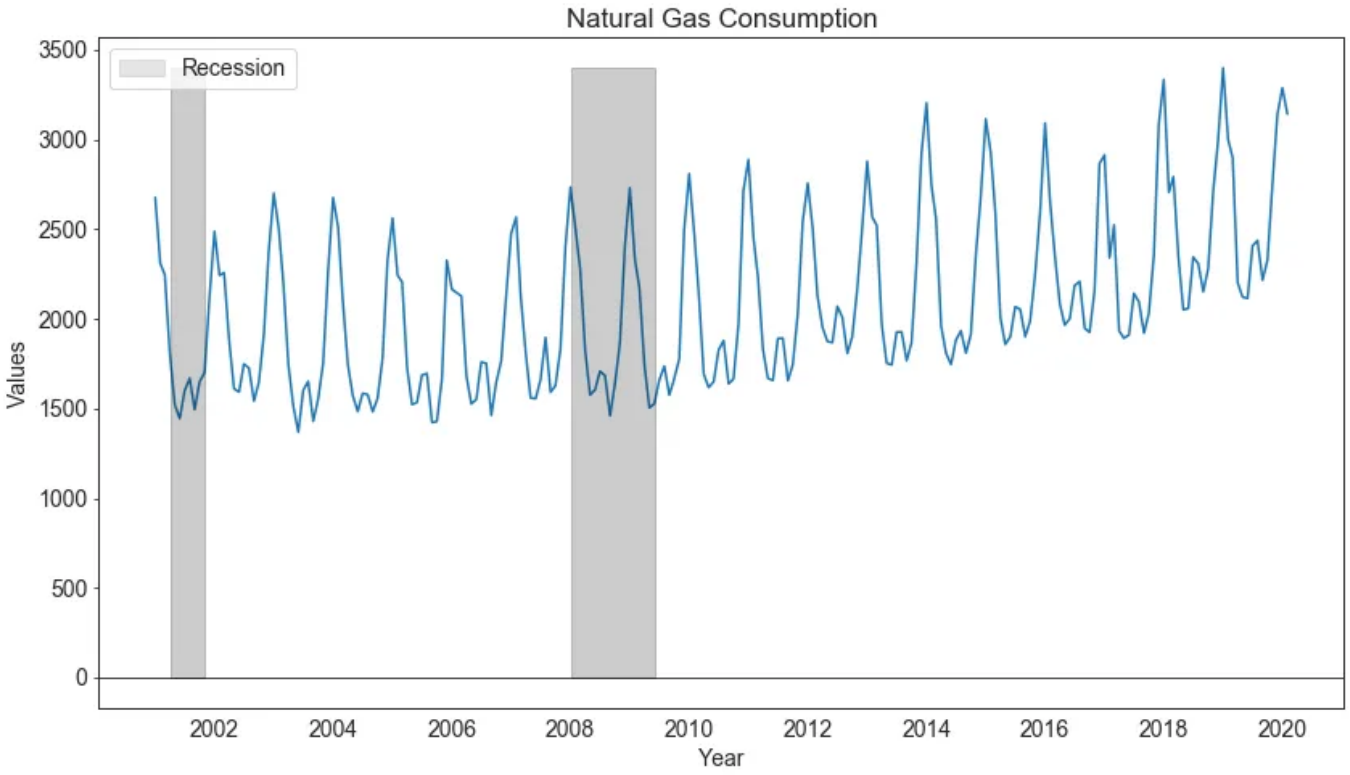
Last time, I wrote extensively about the **ARIMA** model (please read that first), so today I will just cover it at a high level:

- The **AR** stands for **autoregressive** and refers to using lagged values of our target variable to make our prediction. For example, we might use today's, yesterday's, and the day before yesterday's sales numbers to forecast tomorrow's sales. That would be an AR(3) model as it uses 3 lagged values to make its prediction.
- The **I** stands for **integrated**. It means that instead of taking the raw target values, we are differencing them. For example, our sales prediction model would try to forecast tomorrow's change in sales (i.e. tomorrow's sales minus today's sales) rather than just tomorrow's sales. The reason we need this is that many time series exhibit a trend, making the raw values non-stationary. Taking the difference makes our Y variable more stationary.
- The **MA** stands for **moving average**. A moving average model takes the lagged prediction errors as inputs. It's not a directly observable parameter unlike the others (and it's not fixed as it changes along with the model's others parameters). At a high level, feeding the model's errors back to itself serves to push it somewhat towards the correct value (the actual Y values).

S: Seasonality

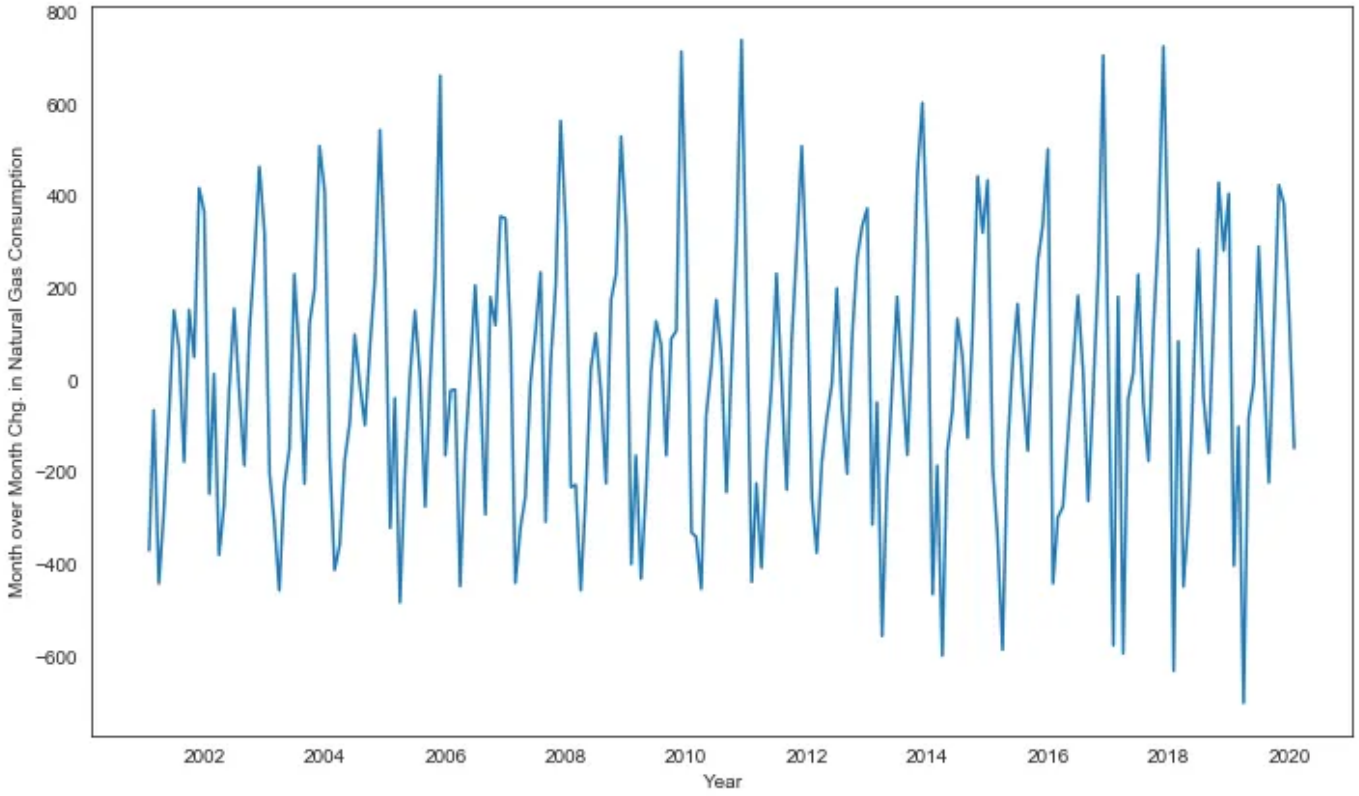
In the intro, we alluded to why seasonality is important — if we know sales spike every winter quarter due to the holidays, then we should account for that. In this case, we definitely would not want to compare 2020 Q1 sales with 2019 Q4 sales without adjusting for this systematic spike — without accounting for seasonality, we would mistakenly assume that sales had cratered (and our company was in big trouble) when it's actually just the expected post-holiday hangover.

Let's take a look at a time series with significant seasonality, natural gas consumption:



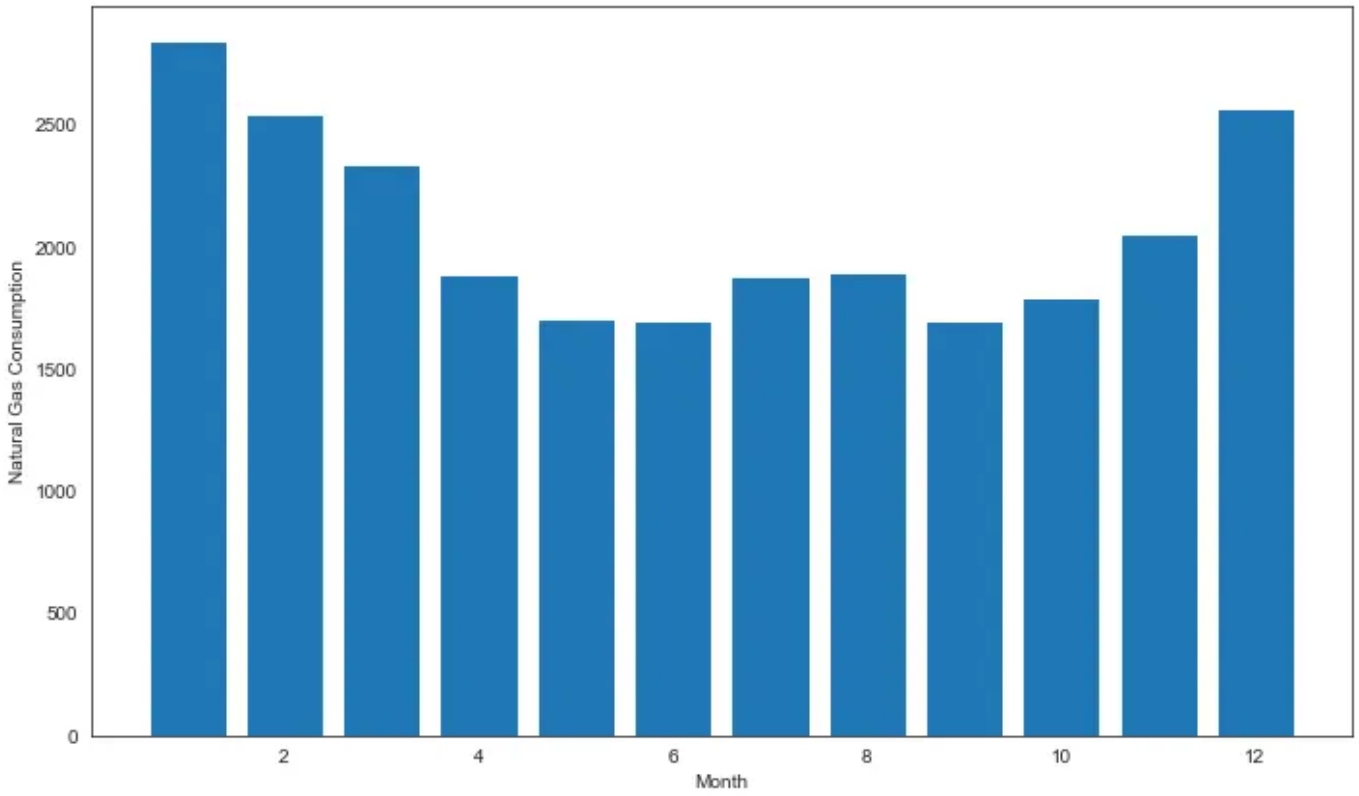
Natural gas consumption (source: Federal Reserve Bank of St. Louis)

Those repeating spikes are evidence of significant seasonality. If we try to graph the month over month changes, we get a similar plot. Again, we see the same approximate sequence over and over again.



Month over month change in natural gas consumption

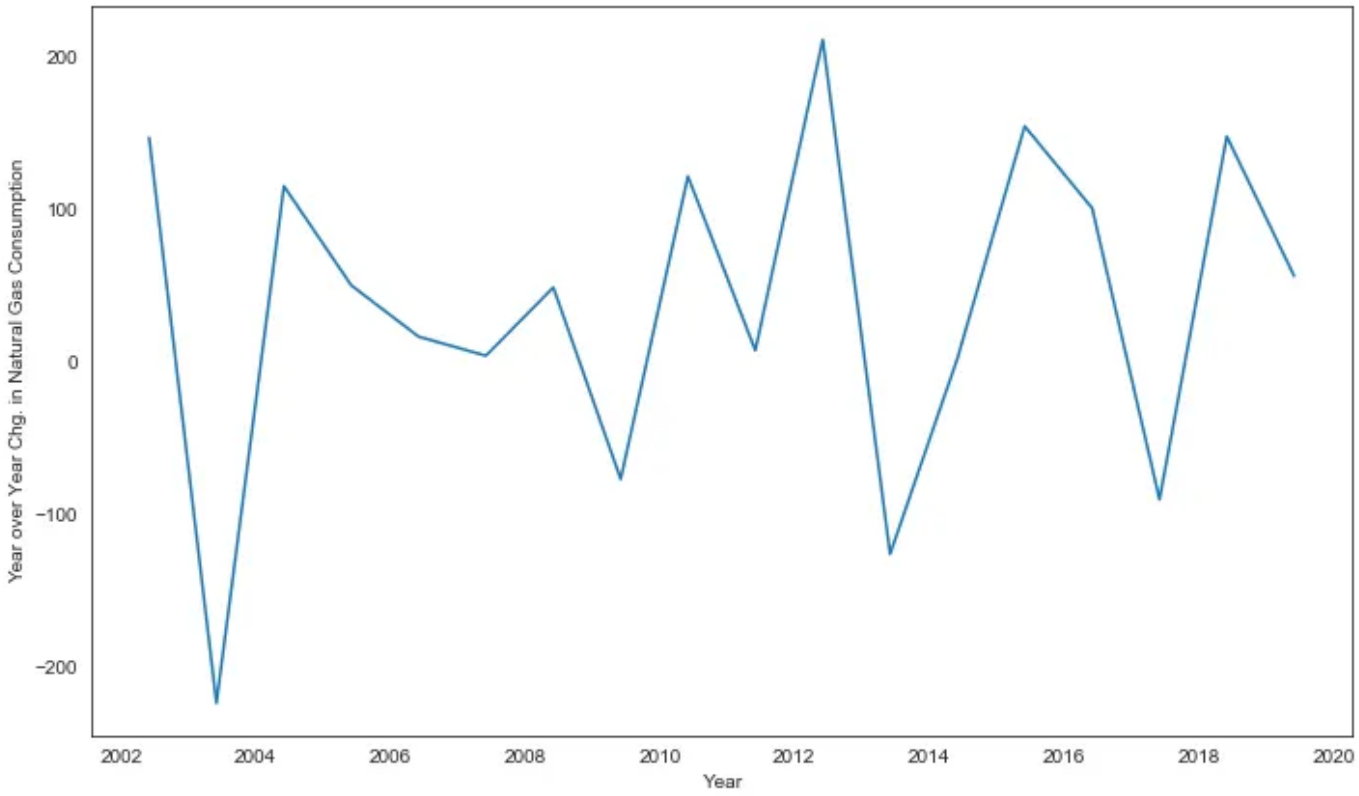
It's consistently cyclical and easily predictable, which means that we should look past the cyclicality (in other words adjust for it). For those unfamiliar with natural gas, the reason consumption spikes and troughs at regular intervals is because one of its primary uses is heating — and demand for heating varies with the seasons (more in the winter months):



Natural gas consumption by month; more consumed during cold months

Let's take a second and think about why I said that we should look past this cyclicality. If we knew that natural gas consumption peaks every January and troughs every June, isn't that enough for a model? If you are building a very simple model, then maybe. But if you want to truly understand the factors behind natural gas consumption, then you need to go a lot deeper than "natural gas usage varies by month".

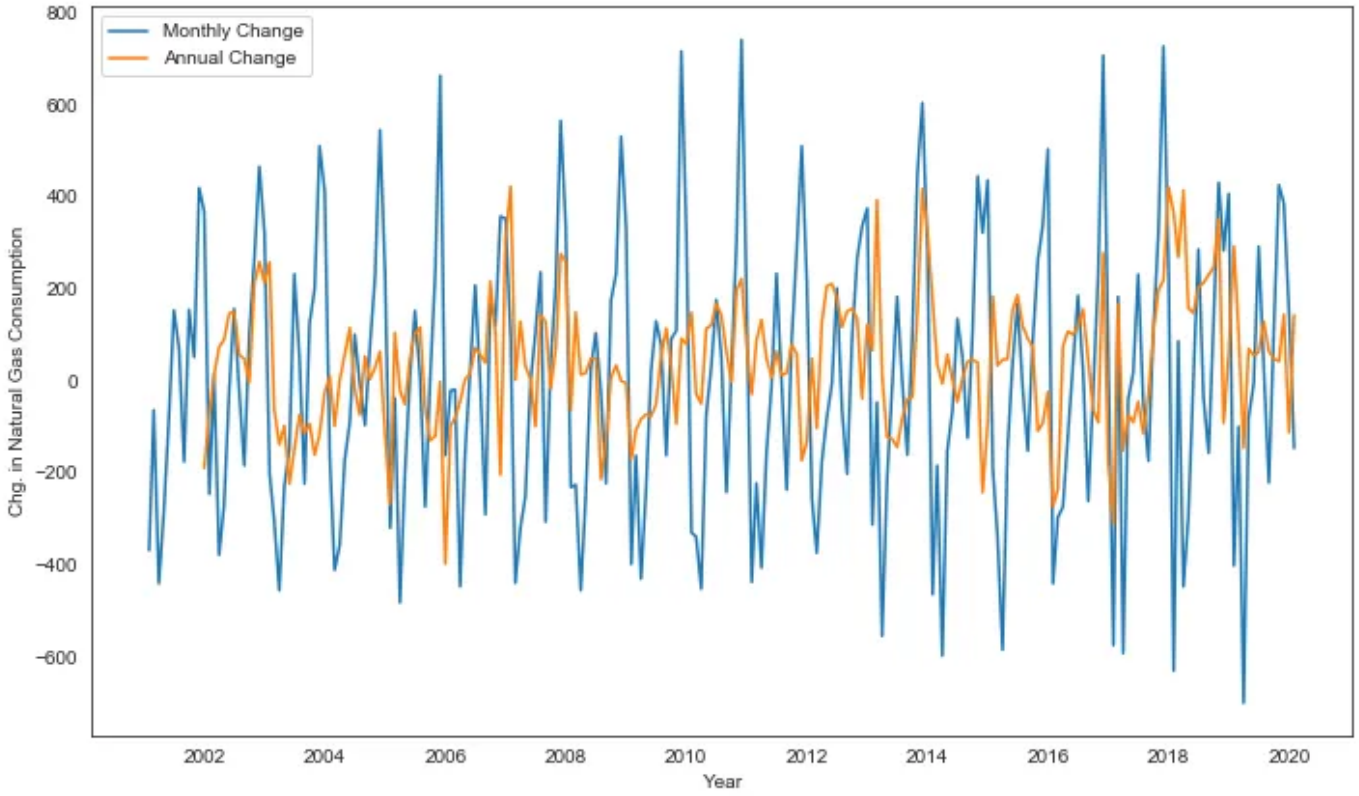
Take a look at the following plot. I sliced my data to include only the data points for June and then calculated the change. So it's a year over year change but for June only — this is a quick way to adjust for seasonality (since we are comparing the same month across years). The drawback is that we are forced to calculate changes at a 1 year horizon, which is quite coarse.



Annual change in natural gas consumption (June data to adjust for seasonality)

The fluctuations in the above chart are potentially more interesting than the previous ones. Since we are comparing just Junes, the variance must be due to factors other than the changing of the seasons such as economic factors, price shocks, or abnormally hot or cold Junes. It's by studying this plot (or similar plots that have also been seasonally adjusted) and correlating its ebbs and flows against exogenous factors that we can **better understand the key drivers of "*beyond the norm*" changes** in natural gas consumption.

In the figure below, I plot both the 1 month and 12 month changes so we can get a sense of the relative magnitudes. Keep in mind that the 1 month change includes seasonality while the 12 month change, like our June to June change, does not:



Comparison of 1 month and 12 month change in natural gas consumption

One thing that becomes apparent from eyeballing the previous plot is that the orange line (12 month change) is a lot less volatile than the blue line (1 month change). This is another clear indicator that seasonality drives much of the observed variance in our data.

Controlling For Seasonality With SARIMA

Often we want to look at higher frequency data such as weekly or monthly changes. Annual changes are great for building long term forecasting models, but using them may cause us to miss out on the granularity necessary for understanding the here and now.

So instead of transforming the Y variable of our regression into an annual change, we can directly control for seasonality by including it as a feature (X variable) in our model. That's what SARIMA does. SARIMA stands for Seasonal Autoregressive

Integrated Moving Average (quite a mouthful). It's very much like ARIMA but more powerful.

We can use statsmodels' implementation of SARIMA. The 3 key arguments for the SARIMAX function are:

- The raw data (stored in a dataframe called gas_df). It looks like this:

```
        date  values
0 2001-01-01  2677.0
1 2001-02-01  2309.5
2 2001-03-01  2246.6
3 2001-04-01  1807.2
4 2001-05-01  1522.4
```

- Similar to ARIMA, the order argument is a tuple that tells the function the number of AR lags, the number of time steps to take the difference over, and the number of MA lagged errors (in that exact order) to include in the regression.
- The argument seasonal_order is similar to order, except it is for specifying seasonality. The first 3 values in seasonal_order are the same as those from order (AR, differencing horizon, MA), but the last value specifies the length of time that defines the seasonality period. So if we believe that the repeating pattern happens over a year (like in our case where differencing from June to June removes the seasonality), we would set it to 12 (because each time step in our data is 1 month long).

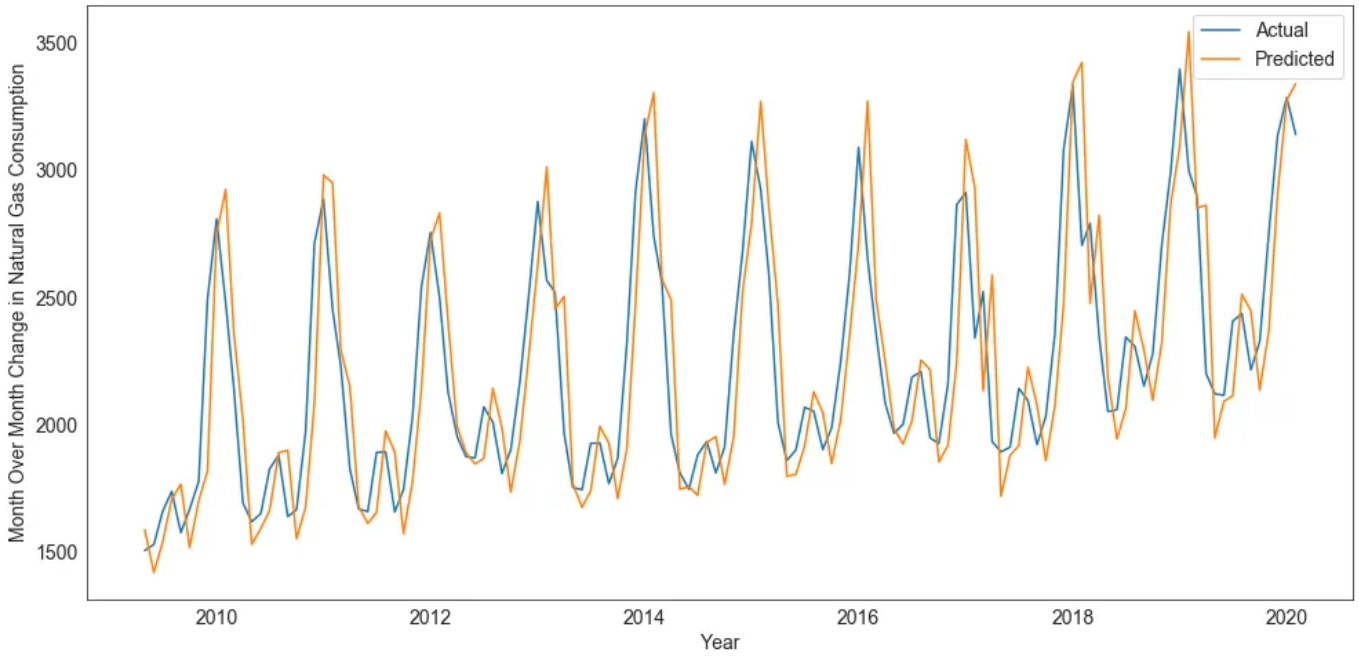
Since this is a toy model for demonstrating SARIMA, I don't do a train test split or do any out of sample stress testing of the model.

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

s_mod = SARIMAX(gas_df['values'],
                order=(1,1,0),
                seasonal_order=(0,0,0,1))
predictions = s_mod.fit().predict()
```

First, let's take a look at how a simple AR(1) model performs on this data (AR(1) means we are making our Y prediction with just the most recent lagged value of the Y variable). In the previous block of code, I set the last value in seasonal_order to 1 (so no special seasonal time step) — this is equivalent to an AR(1) model as I removed the MA and seasonal components.

Notice how the prediction (orange line) always seems a step slower than the actual (blue line). This is a classic problem of AR models — when you have only the past to go on, then radical shifts in the trend become nearly impossible to predict.



AR(1) prediction

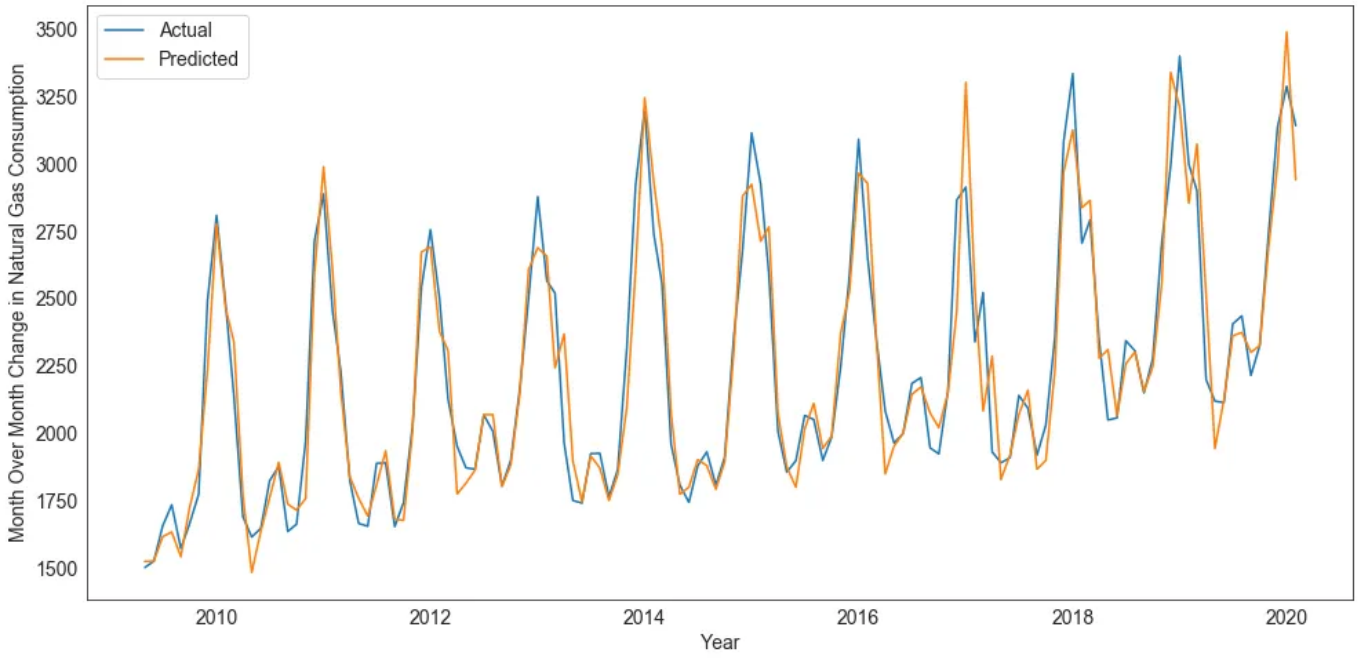
Since we observed regular, annual cycles in natural gas consumption, let's see if adding in the seasonal component improves things. We do this by changing the seasonal_order argument to (1, 0, 0, 12) — this gives us one seasonal lag, meaning we use the value from one year ago (12 lags) to make our prediction. Note that to isolate out the impact of seasonality, I removed the AR lag in the order argument.

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

s_mod = SARIMAX(gas_df['values'],
                order=(0,1,0),
```

```
                    seasonal_order=(1,0,0,12))
predictions = s_mod.fit().predict()
```

Here is the SARIMA prediction with seasonality accounted for. By the way, one nice thing about SARIMAX relative to ARIMA in statsmodels is that the output of the predict method is the predicted value of the target variable itself. Whereas for ARIMA, the output of the predict method is the predicted change in the target variable (so you need to do an extra adjustment to get back the raw value).



SARIMA prediction

Notice the much better fit. The orange line no longer lags; it now tightly overlays the blue line. Great!

FYI for the curious, we can actually obtain a similar result with an AR(12) model. An AR(12) includes enough lags to capture the seasonality of the time series, but requires us to estimate an additional 11 betas (the betas for lags 1 through 11) so it's more prone to overfitting.

Putting It All Together

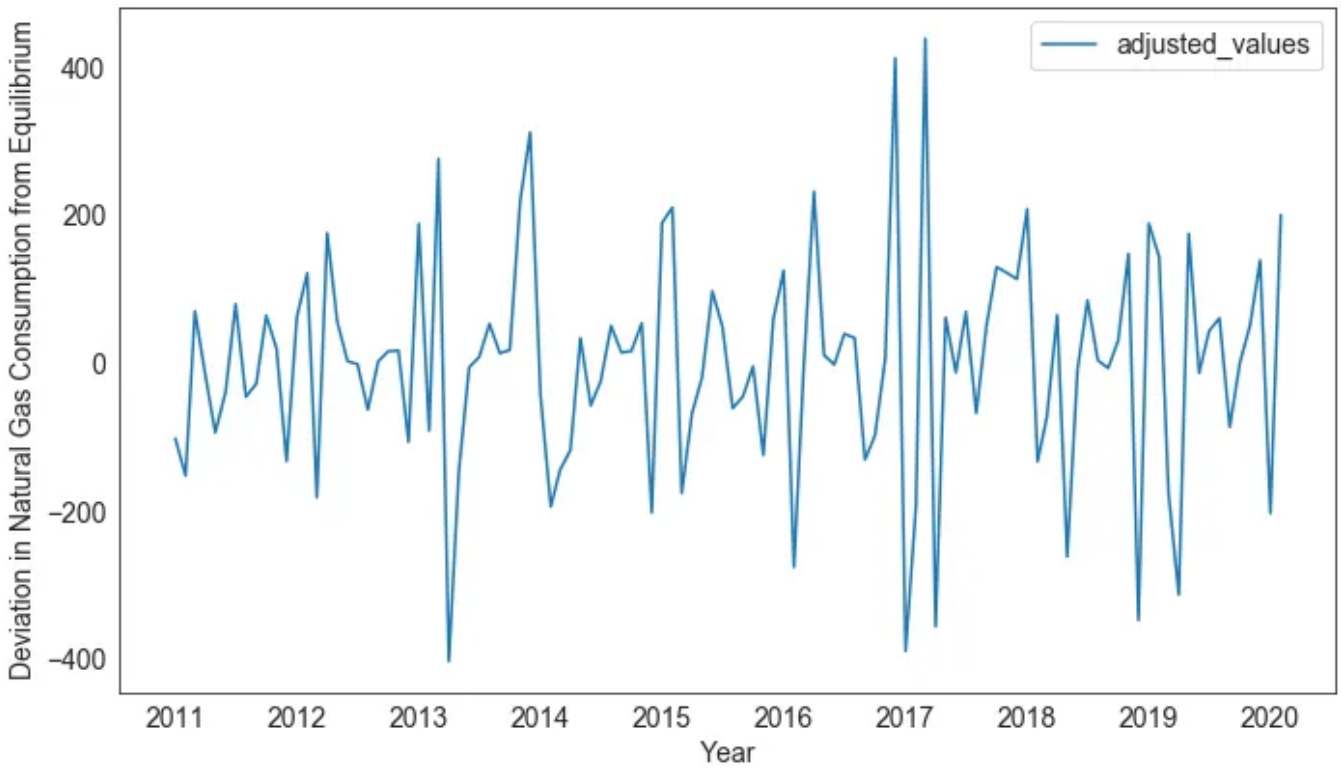
Cool, the lines overlap (meaning that our SARIMA prediction overlays the actual thanks to the inclusion of seasonality). So what?

Well, we can do one of 2 things. Either we can add other factors to our SARIMA model in an attempt to explain the residual (unexplained) variance or we can create a seasonally adjusted series, in other words, a new Y variable. Both of these are similar in spirit, but I generally prefer the second way. I like to plot and look at time series; it helps me understand them.

So let's go with the latter approach. We can create a seasonally adjusted version of natural gas consumption by subtracting the output of our SARIMA model from the actual observed value (blue line minus the orange line).

```
gas_df['adjusted_values'] = gas_df['values'] - predictions
```

This works because the output of our SARIMA model, where the only factor was seasonality, can be thought of as "the level of natural gas consumption where seasonality is allowed to vary but all other things are held constant". So any deviations from this equilibrium seasonal level must be due to other factors. Here is what the seasonally adjusted one looks like:



Actual less seasonal equilibrium natural gas consumption

We can check that this works by comparing the adjusted values against a time series that we know is relatively free from seasonality, the annual change in natural gas consumption. The following table shows the correlations between the 12 month change , the 1 month change (impacted by seasonality), and our adjusted value:

	gas_chg12	gas_chg1	adjusted_values
gas_chg12	1.000000	0.199713	0.506134
gas_chg1	**0.199713**	1.000000	0.511884
adjusted_values	**0.506134**	0.511884	1.000000

Notice the much higher correlation (0.5 vs. 0.2) between the adjusted value and the 12 month change. It's fine that the correlation is not closer to 1 because the 12 month change is a somewhat coarse way of adjusting for seasonality, so we don't want to reproduce it exactly.

Cool, now we know that if we find repetitive, seasonal patters in our time series data, we can use SARIMA to account for it. Good luck forecasting! Cheers!