ABSTRACT

In this final, I picked my own mathematical form of model based on the data and the preliminary visualization given by the professor and tried to optimize it with Bayesian methods. I discussed the preprocessing of data, the likelihood model, the factor graph representing the model, prior distribution, the convergence of the Rstan algorithm, and the prediction of CO_2 in 2058.

MODEL

Pre-processing

Noted that the date of the dataset is not continuous and mostly marked every week from 1958 to 2017. I manually transform it in Excel so that each date is transformed into a timestep correspond to a day. For example, date 1 in the dataset has a time step 1, while date 2 has timestep 7.

Likelihood / Model

The model I am using is

$$p(x_t|\theta) = N(c_0 + c_1 t + c_2 \cos(\frac{2\pi\, t}{365.25+phase})\ +\ c_3 t^2,\ \sigma^2)$$

In which

- Long-term trend: quadratic, $c_0 + c_1\, t +\ c_3 t^2$
- Seasonal variation (every 365¼ days): cosine, $c_2 \cos(2\ \pi\ t\ /\ 365.25 + phase)$
- Noise: Gaussian with 0 means and fixed standard deviation, c_4
- The c_i variables are all unobserved parameters of the model.

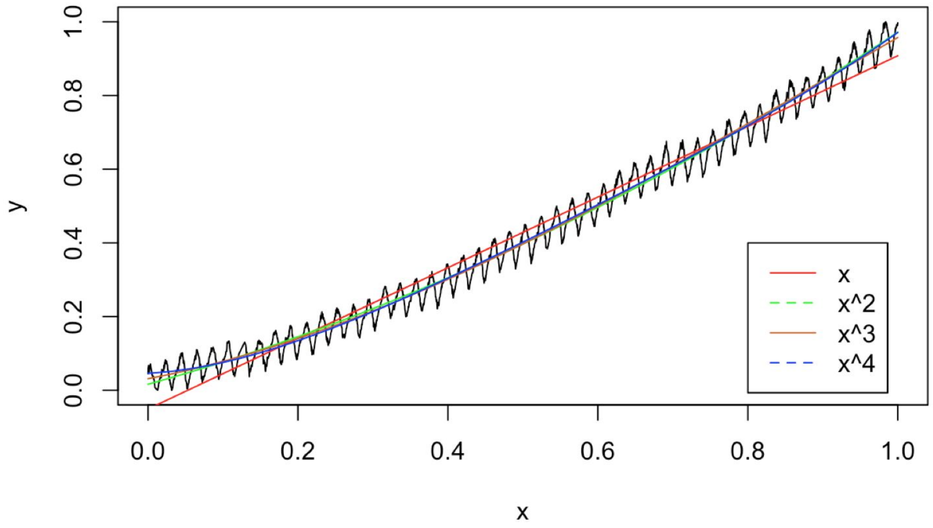
The assumption of the model includes:

1) Seasonal variation is unchanged over time, which means treating seasonal variation as a stable periodic function
2) There is no interference of unstable noise, which means noises are normally distributed, without the interference of measurement errors etc.

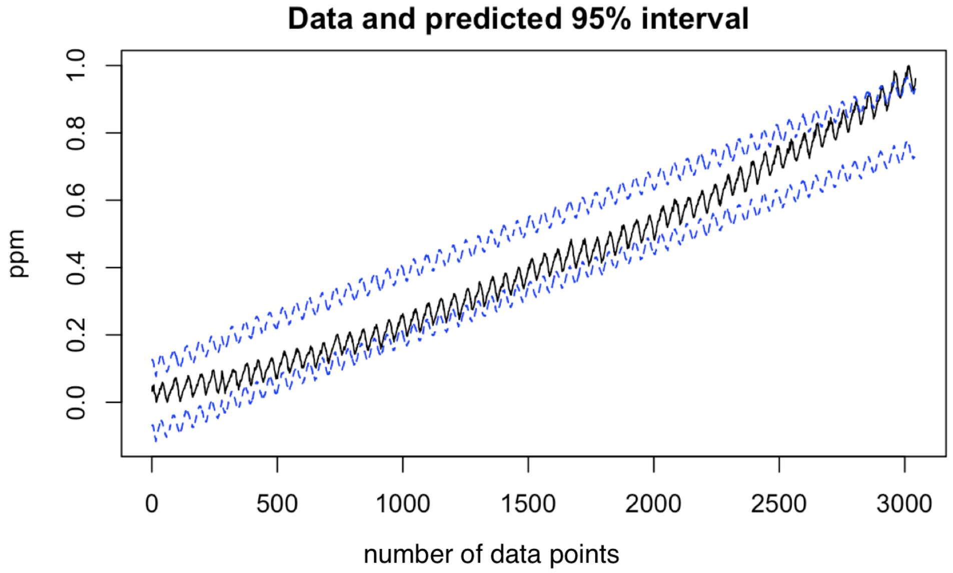
I only have time to exam a better trend line parameters. I tried fitting different order of polynomial regression on the dataset and it seems like quadratic, cubic, quartic regression both fit the data pretty well. So I just picked adding a quadratic term to the trend line to keep things simple.

I understand that to seek for a better model, I could have looked at a different seasonal variation function and a different noise function. One way to get better period function is to first isolate the period component from the function and just compare data to our period function. Try tweaking period parameters to see if it gets closer to capture the shape of the data.

To get a better noise model, I can calculate the difference of values between two seasonal lumps, which essentially control for the effect of seasonal variations. The seasonal trend might be small enough to be ignored or to be safe we can also subtract it from the difference. Then we can see if the noise distribution is normally distributed and seek for a better model.

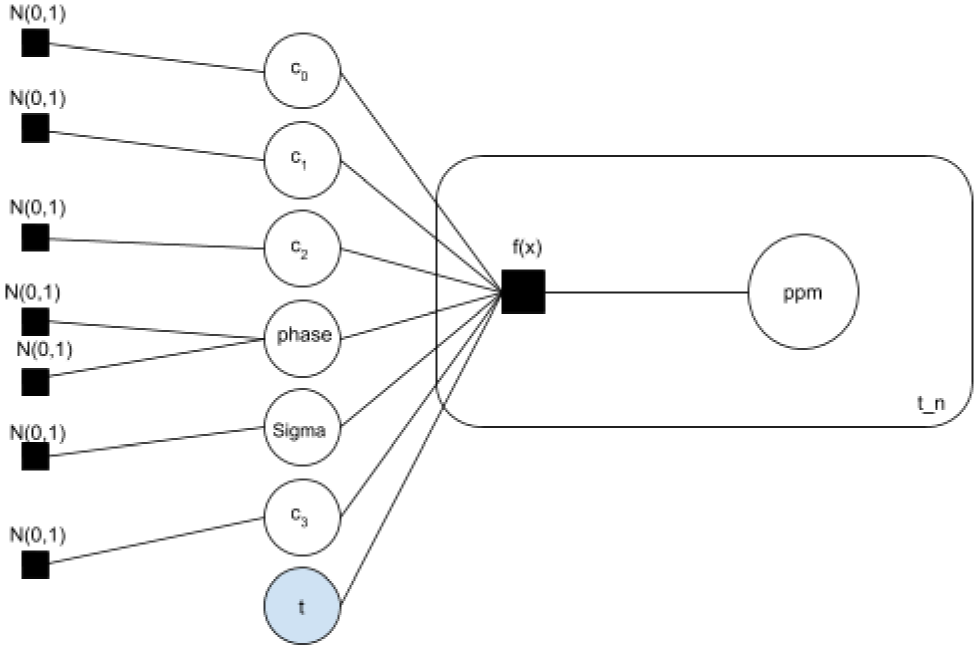


(Figure 2: Fitting different polynomial functions)



(Figure 3: Predicted 95% interval based on the model professor provided. This implies that a quadratic function in needed to better accommodate the model).

Factor Graph Representing the Model



(Figure 4: Graphical model)

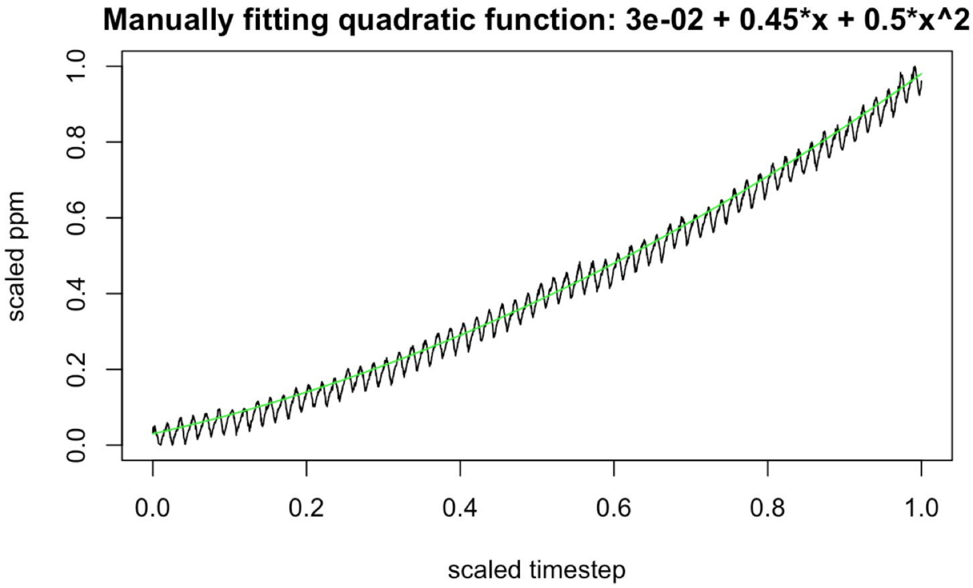
Prior

The current prior distribution for all parameters are almost the same (N(0,1) as the uninformative priors because ppm values are scaled between 0 and 1) and all are constrained to be positive numbers through transformed parameters, except that phase parameter is constrained to ($\pi, 2\pi$).

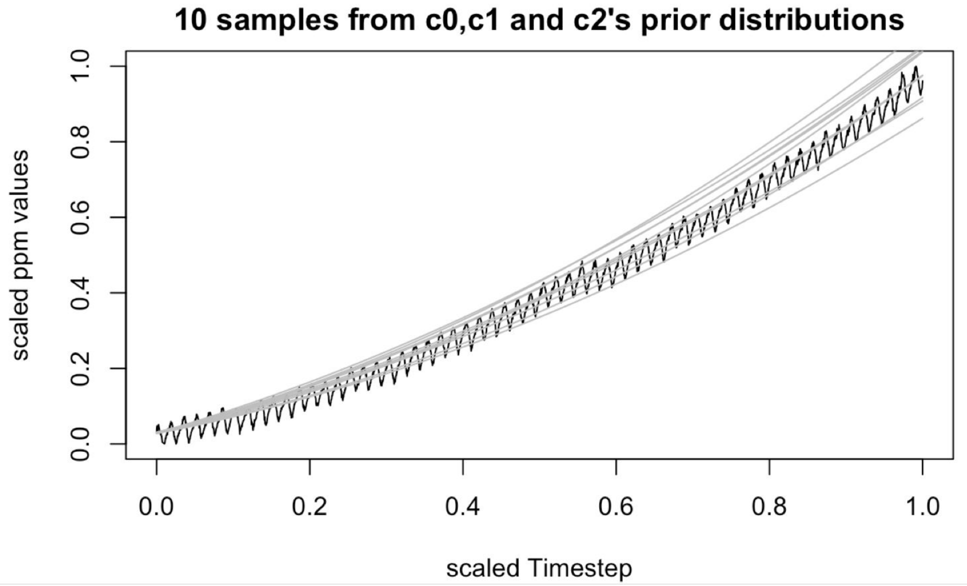
```
transformed parameters {
real<lower=0> c0;
real<lower=0> c1;
real<lower=0> c2;
real<lower=0> phase;
real<lower=0> sigma;
real<lower=0> c3;

c0 = exp(c0_prime);
c1 = exp(c1_prime);
c2 = exp(c2_prime);
phase = (atan2(phase_x,phase_y)+pi());
sigma = exp(sigma_prime);
c3 = exp(c3_prime);
}
```

```
model {
//Priors
c0_prime ~ normal(0,1);
c1_prime ~ normal(0,1);
c2_prime ~ normal(0,1);
phase_x ~ normal(0,1);
phase_y ~ normal(0,1);
sigma_prime ~ normal(0,1);
c3_prime ~ normal(0,1);
```

I have tried very hard to find better prior distributions for each parameter, but my model performs better with uninformative priors. The reason is probably that I have not scaled certain parameters/values correctly or didn't translate correctly between prime_parameters and parameters. At first, I tried to find better priors for trend line coefficients. I scaled both ppm values and timesteps to between 0 and 1 and try to manually fit a quadratic regression function to the data to extract some prior coefficients.



(Figure 5: The quadratic function I fit with c0 = 3e-02, c1=0.45 and c3 = 0.5)



(Figure 6: 10 samples from trendline priors)

c0 = c(rnorm(10, 3e-02, 1e-03))
c1 = c(rnorm(10, 0.45, 0.05))
c3 = c(rnorm(10, 0.5, 0.05))

(10 samples from c0, c1 and c3's prior distribution, with normal distribution defined around the fitted values found above).

However, when I tried to use the new priors in the model, I am not sure how to transform the mean and variance for their corresponding prime parameters. For example, c0 = exp(c0_prime) and c0_prime becomes the prior we actually need to define. I tried just logging the values for c0 (e.x. c0_prime ~ N(-3.5, -6.9)), but the model doesn't allow negative number even though the constraints I put is only on c0. I also tried calculated c2's prior distribution manually by looking at the amplitude of data. It was within the range [0.01, 5] ppm, so what I want is a distribution for exp(c2_prime) so that 95% of its mass is between [0.01, 5]. I will first normalize it by logging it and then find the appropriate mean for normal distribution as the mean of the interval, through which I deduce std. c2_prime prior ends up drawing from N(-0.6505, 0.67475). Anyhow, no matter how I try with different prior, they mass up my model more than the uninformative uniform prior. Most likely it's because I didn't scale down other parameters (e.g. such as phase) correctly in the process.

So in the following prediction, I all use the uninformative priors.

INFERENCE + CONVERGENCE

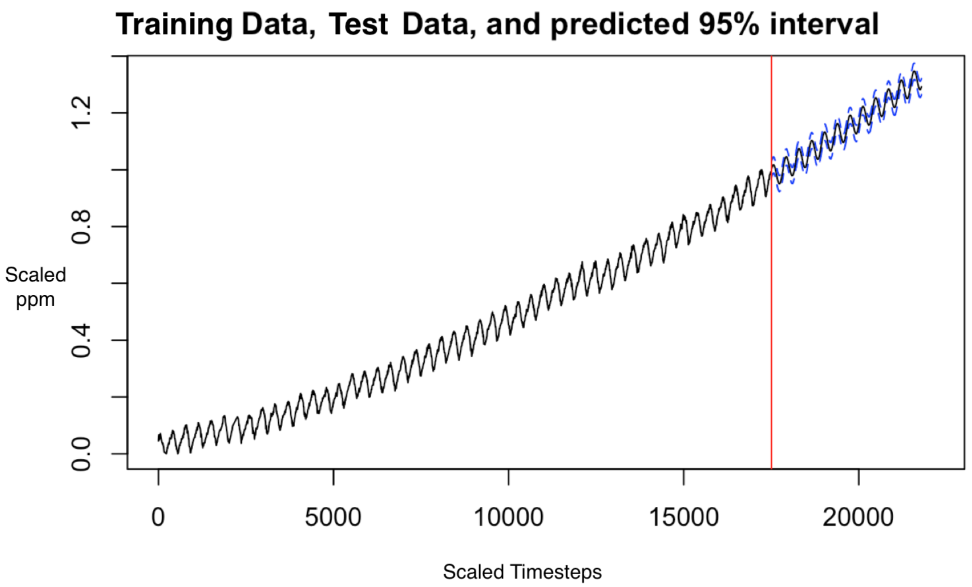
What Rstan outputs are three things:

1) A table from which we can see the inferred parameter values. N_eff and Rstan tells us whether the algorithm converges and how much we can trust the inference Rstan produces. The higher the effective number of samples the better. Low n_eff usually means samples are highly correlated. Rstan around 1 indicates inference from Rstan is solid.
2) From the Rstan fit, we can extract the posterior distribution of each x value, separated in quantiles, to do prediction.

In this case, priors are less important as long as they are in a reasonable range and constraints because we have a large amount of data. The Rstan model takes in a defined model (which I improved based on professor's model), calculate the likelihood accordingly with data input (time steps and ppm) and eventually time it with priors. Here we not only infer the parameters' values, but we also use those values to predict for ppm in 2058.[1]

EVALUATION OF THE MODEL

While running the model and trying to set priors, I have only run them on a training set, which is 80% of the dataset. To evaluate the model, I ran RMSE on the predicted data from the model and the actual data (test set), which gives the value of 392.4088. Even visually speaking, it's a much better model fit than the baseline model provided by the professor. I could have done more rigorous test statistic, but I am running out of time for model checking and comparison.

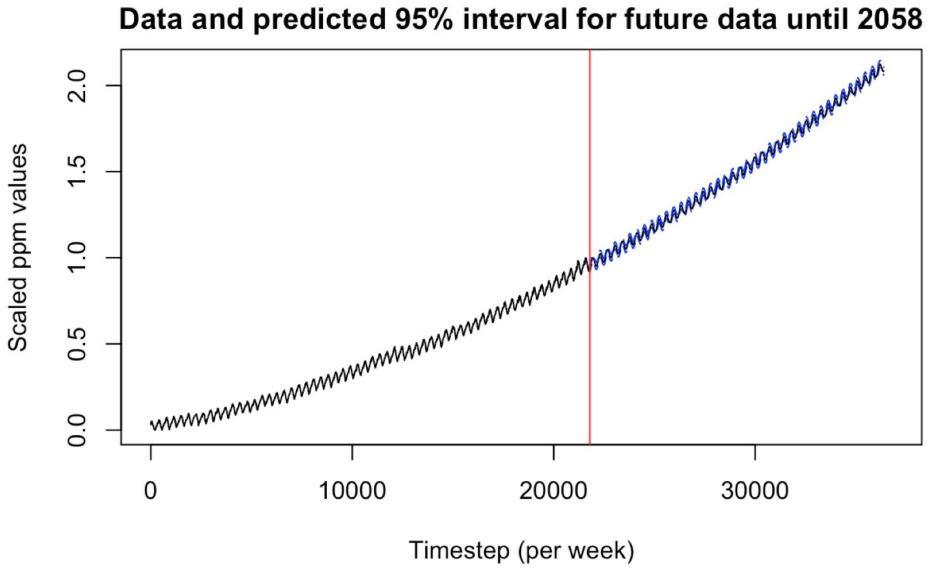


(Figure 7: Predicted interval on test data. Redline is the separation between train and test data)

[1] I would have analyze inferred parameter values a bit more if my R markdown file didn't lose all my progress and graph in the last min… Now I cannot even retrieve the table (because that require me to rerun the Rstan model).

RESULT

My estimate for atmospheric CO_2 levels projected until the start of 2058 is 515.5992ppm. The 95% confidence interval is [513.5645, 517.6664]



(Figure 1: Predicted confidence interval for future data until 2058).