Statistical Computation Description

Computational statistics is the use of computer science to implement statistical computations, one case of which is statistical data analysis. In this paper, a Bayesian model is built to project the rise of $CO^2\ ppm$ (parts per million) emissions between the year 2017 and 2058, using weekly data from 1958 through 2017, collected via the Mauna Loa Observatory in Hawaii. The benefit of using a statistical model in this instance is to understand how reliable a single point estimate is – nothing is simply a number, but rather a distribution with a measure of uncertainty. The dataset from 1958 through 2017 is plotted below, demonstrating the overall trends of CO^2 in the atmosphere, providing us with a detailed look at how to approach the modeling.

Model

The likelihood is a function that ranges over all values of N – the number of weeks in the dataset – and takes the form of a normal distribution with 5 transformed parameters. There are three components to model – noise, seasonal variation, and trend.

Noise: The $c4\_transf$ parameter below is modeling the standard deviation of the function, roughly how much variation there is in a single time-step.

Seasonal Variation: The amount of CO^2 in the atmosphere ranges over seasons, due to the life cycle of plans – shedding their leaves in winter allows them to decompose and release carbon dioxide, while regrowth is the most natural form of carbon scrubbing come May. Rob Monroe explains it quite poetically as, barren winter branches that turn to bountiful spring leaves (Monroe, 2013). To model this sinusoidal seasonal variation, a cosine function is implemented, with the start determined by the parameter $c3\_transf$ and the scale determined by $c2\_transf$, with a cycle determined by the length of a year.

Upward Trend: The parameter $c0\_transf$ dictates where the predictions start, intersecting what is the y-axis of where the model starts. To model the upward trend the first implementation included $c1\_transf * t[n]$, where $c1\_transf$ is the slope. In an attempt to model the nonlinearity of the model in a quadratic function, the term $c5\_transf * t[n] * t[n]$ is used.

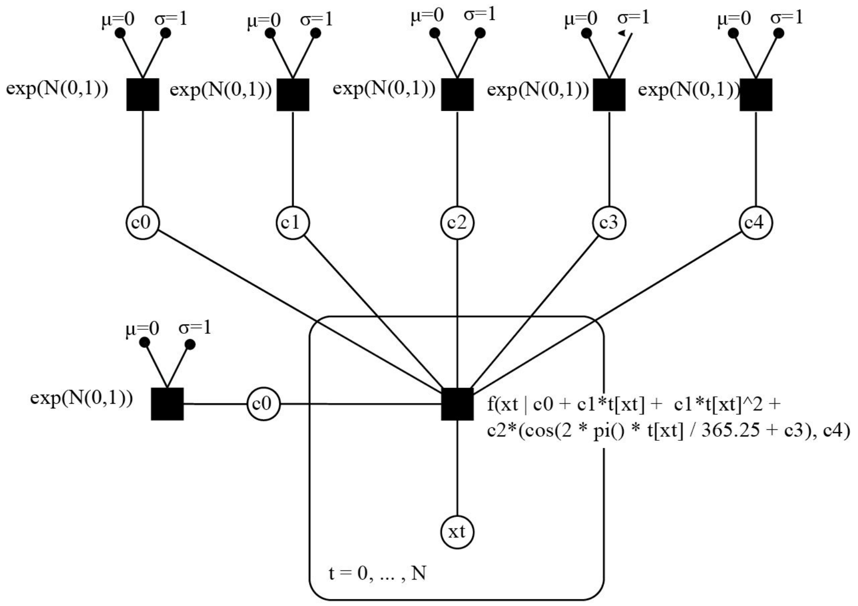
Parameters and Transformations

Parameters $c0,\ c1,\ c2,\ c3,\ c4,$ and $c5$ are used, described above for their interactions within the likelihood. In an attempt to unbound each parameter the samples are exponentiated, so as to force each value to be positive. This relies on the assumption that each parameter will indeed be positive, which seems fair in this situation – $c0$ (intersection will be >350 ppm), $c1 + c5$ (positive sloping), $c2$ (scale of the cosine function), $c3$ (number between 0 and 2π for the displacement of the cosine function), and $c4$ (variance of the model).

Priors over Parameters

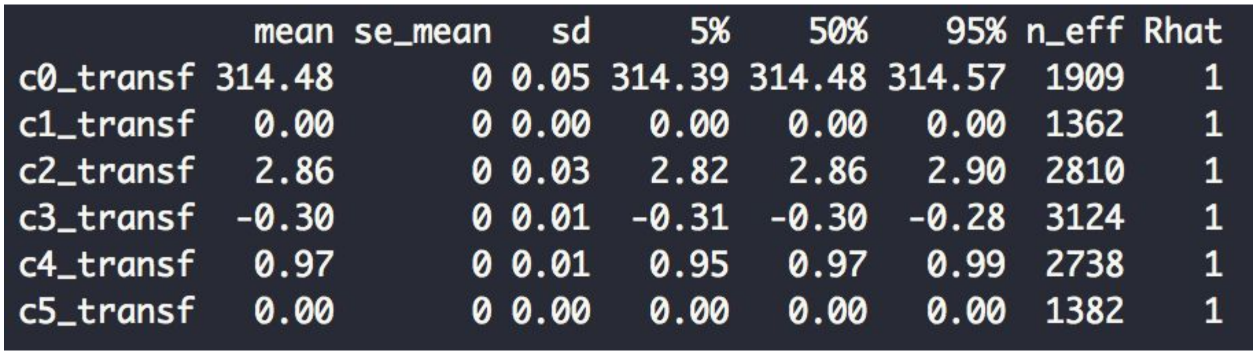
In this model, the priors over the parameters were very uninformative, and the values were less important than the transformations. To set everything equal to roughly the same starting value, all priors are set to normal distributions with mean equal to zero and standard deviation equal to one.

Factor Graph


μ=0 σ=1
exp(N(0,1))
c0
c1
c2
c3
c4
f(xt | c0 + c1*t[xt] + c1*t[xt]^2 + c2*(cos(2 * pi() * t[xt] / 365.25 + c3), c4)
xt
t = 0, ... , N

Convergence and Samples

To measure convergence of the model, Rhat must be close to 1 – often should equal 1 – which it does for every parameter in this model. The n_eff is another useful metric, if Rhat = 1, to see if there were enough effective samples. Enough samples, defined as ___, indicates there is a lower standard error of the mean and more stable estimates. This model again shines through with a large number of effective samples, always >1300.

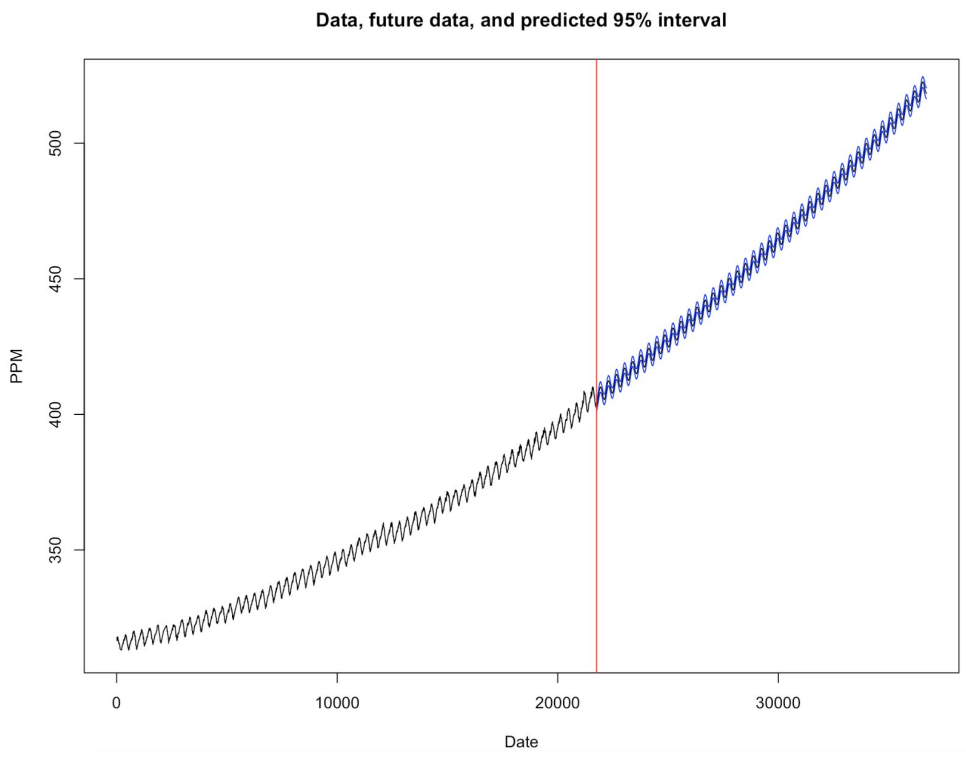


```
              mean se_mean   sd     5%    50%    95% n_eff Rhat
c0_transf 314.48       0 0.05 314.39 314.48 314.57  1909    1
c1_transf   0.00       0 0.00   0.00   0.00   0.00  1362    1
c2_transf   2.86       0 0.03   2.82   2.86   2.90  2810    1
c3_transf  -0.30       0 0.01  -0.31  -0.30  -0.28  3124    1
c4_transf   0.97       0 0.01   0.95   0.97   0.99  2738    1
c5_transf   0.00       0 0.00   0.00   0.00   0.00  1382    1
```

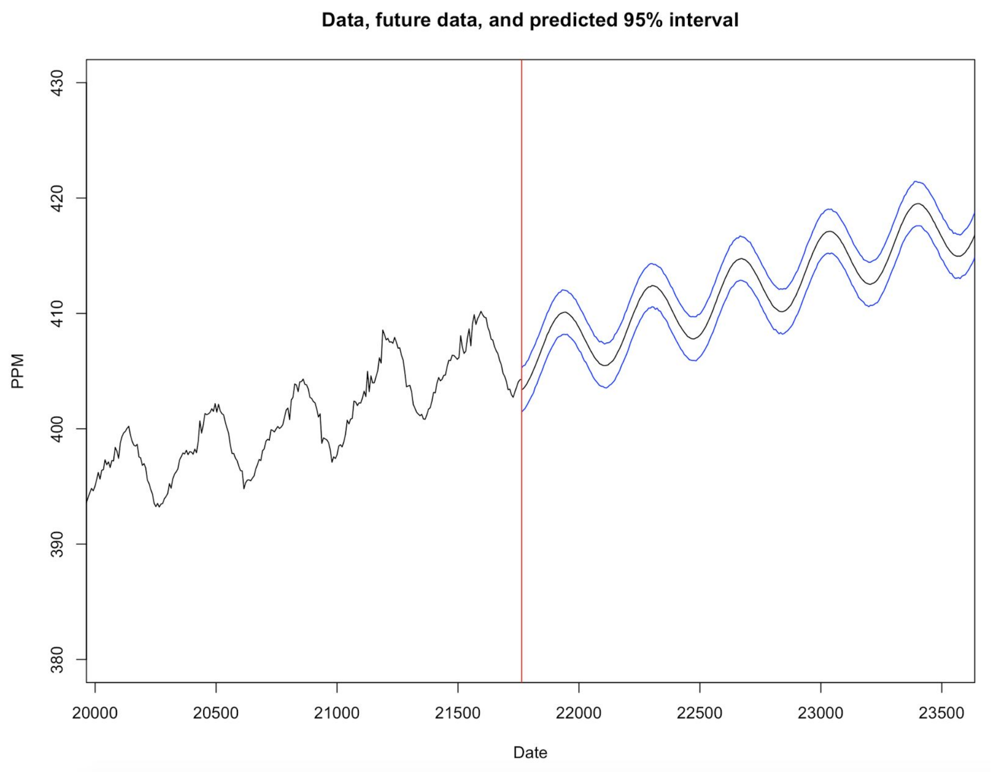
Results

CO2 levels in the atmosphere are currently around 410 ppm (December 2017), projected to be 522.5877ppm in the year 2058. Many scientists agree 450 ppm is a dangerous level for CO2, and although we might be making progress toward a greener world, my guess is that we don't have much we can do to slow this trend now.

The graphs below show the data in black, the separation of training and predictions in red, the 95% confidence intervals in blue, and the projected data in black after the red line. *Graph 1* shows all of the available data, and the projections from today until 2058, giving a clear view of the three components mentioned above – noise, seasonal variation, and long term trends.



Graph 1



Graph 2

Inference and Complications

Stan is a pretty finicky model, one that requires a lot of thought before entering the stage of running on 4 cores on your computer. Every time I ran into an issue, I went through every single line of code and tried to decipher where my model was not working in the way I intended it to train. One error was making the prior for c0 = Normal(350, 1) to model the information I know about the datas intersection with y-axis. The error in this is that I proceeded to exponentiate the parameter, forcing my model to throw a fit and find the most absurd R_hatts I've ever seen. This is one example of a time that my model didn't converge because I moved one piece of a variable.

References

Monroe, R. (2013, June 04). Why Does Atmospheric CO2 Peak in May? Retrieved December 16, 2017, from https://scripps.ucsd.edu/programs/keelingcurve/2013/06/04/why-does-atmospheric-co2 peak-in-may/

Code

```
# Load data
setwd("/Users/Colette/anaconda/1CS 146/FINAL/")
raw.data <- read.csv('co2.csv', header = FALSE)

df <- subset(raw.data, select = c('V1', 'V2', "V3", "V4"))
colnames(df) <- c('date', 'ppm', 'raw_date', "scaled_ppm")

# removing weird NA's
df <- df[-c(1, 3041, 3040), ]

# viewing the data to inspect the trends
plot(df$date, df$ppm, type = "l",
    xlab="Time", ylab="PPM",
    main = "Mauna Loa CO2 (PPM)")

t_future <- seq(max(df$date), (max(df$date) + (2058-2017)*365), by = 7)
predict <- data.frame(t_future)

# defining data values
N = 3038             # 3038 data points for training
n_future = 2138  # 2138 data points to predict
t = df$date          # dates/timesteps for training
t_future = predict$t_future  # dates/timesteps for predictions
ppm = df$ppm     # ppm

# setting the data for Stan
stan_data <- list(
  N = N,             # 3040 data points
  t = t,             # dates/timesteps
  ppm = ppm,         # ppm
  n_future = n_future,       # number of weeks
  t_future = t_future
)
```

# **Stan!**

```
stan_model = "
data {
int <lower = 0> N;                // the number of timesteps in the data
int <lower = 0> n_future;         // number of future timesteps to be predicted
vector [N] t;                     // timesteps
vector [n_future] t_future;       // future timesteps
vector [N] ppm;                   // ppm values
}

parameters {
real c0;
real c1;
real c2;
real c3;
real c4;
real c5;
}

transformed parameters {
real c0_transf;
real c1_transf;
real c2_transf;
real c3_transf;
real c4_transf;
real c5_transf;

c0_transf = exp(c0);
c1_transf = exp(c1);
c2_transf = exp(c2); // keep this for sure (mapping real numbers to positive real)
// phi_transf = (atan2(phi_1, phi_2)); // phase between 0 and 2pi (think logistic function
(2pi()/1+e^-x))
c3_transf = (c3);
c4_transf = exp(c4);
c5_transf = exp(c5);
}

model {
// Priors
c0 ~ normal(0, 1);
c1 ~ normal(0,1);
c2 ~ normal(0,1);
c3 ~ normal(0,1);
c4 ~ normal(0,1);
c5 ~ normal(0,1);

// Likelihood
for (n in 1:N)
  ppm[n] ~ normal(c0_transf + c1_transf * t[n] + (c5_transf * t[n] * t[n]) +c2_transf*cos(2 * pi() *
t[n] / 365.25 + c3_transf), c4_transf);
}
```

```
generated quantities {

  real ppm_future[n_future];
  for (x in 1:(n_future)){
    ppm_future[x] = normal_rng(c0_transf + c1_transf * t_future[x] + (c5_transf * t_future[x] *
t_future[x]) + c2_transf*cos(2 * pi() * t_future[x] / 365.25 + c3_transf), c4_transf);
  }
}
"

library(rstan)

fit <- stan(
  model_code = stan_model,
  data = stan_data,       # named list of data
  chains = 4,             # number of Markov chain
  warmup = 1000,          # number of warmup iterations per chain
  iter = 2000,            # total number of iterations per chain
  cores = 4,              # number of cores
  refresh = 1000,         # show progress every 'refresh' iterations
  control = list(adapt_delta = 0.9)
)

samples <- extract(fit)
print(fit, par=c('c0_transf','c1_transf','c2_transf','c3_transf','c4_transf', 'c5_transf'), probs = c(0.05,
0.5, 0.95))

results <- apply(samples$ppm_future, 2, quantile, probs = c(0.025, 0.5, 0.975))

print(all_time[min(which(results[2,1:n_future] * sd(data$CO2) + mean(data$CO2) > 450))])

# Plotting the entire dataset and all predicted values
plot(
  df$date, df$ppm,
  col='black', type = 'l',
  xlim=c(0, t_future[n_future]),
  ylim=c(min(c(results[1,], ppm)), max(c(results[2,], ppm))),
  xlab="Date", ylab="PPM",
  main='Data, future data, and predicted 95% interval')
lines(c(predict$t_future), c(results[1,]), col='blue')
lines(c(predict$t_future), c(results[2,]), col='black')
lines(c(predict$t_future), c(results[3,]), col='blue')
```

```
abline(v=t[N], col='red')

# Zooming in on the plot of the entire dataset and all predicted values
# Inspect this to see how well min(future_ppm) matches max(ppm)
plot(df$date,df$ppm,xlim=c(20100,23500),ylim=c(380,430),
    col='black',type="l",
    xlab="Date", ylab="PPM",
    main='Data, future data, and predicted 95% interval')
lines(c(predict$t_future), c(results[1,]), col='blue')
lines(c(predict$t_future), c(results[2,]), col='black')
lines(c(predict$t_future), c(results[3,]), col='blue')
abline(v=t[N], col='red')

# Print the ppm in the year 2058
projected_ppm <- max(results[2,])
print(cat("CO2 ppm is projected to be", projected_ppm, "in the year 2058"))
```