STAT 447C Project: Bora Guney and Damien Fung

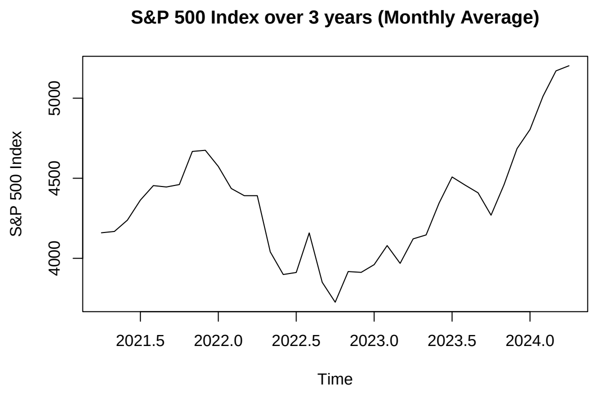
2024-04-09

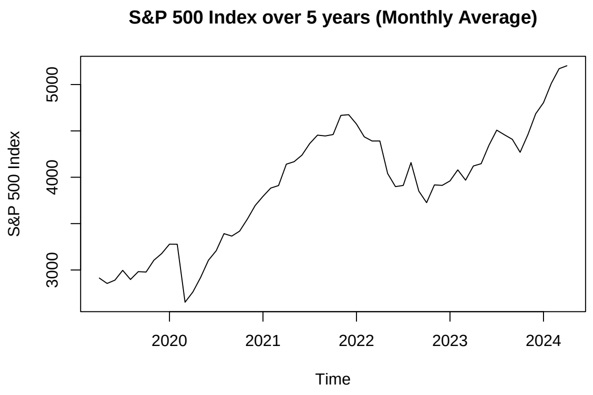
Introduction

Throughout the duration of this outgoing academic term, we have both been undergoing STAT 443: Time Series & Forecasting. Taking both STAT 443 and STAT 447C in tandem, we have grown a keen interesting in the utilization of Bayesian techniques within time series forecasting. Namely, this project aims to compare parameter estimates and prediction performance obtained using Bayesian techniques compared to those obtained using ARIMA. To demonstrate this, we utilize the S&P 500's historic data from the previous 3 years. We develop our Bayesian model using MH MCMC to learn model parameters for the 3 year data set. Furthermore, we employ a hierarchical model where we treat the 5 year data set as side data to inform our prior choices. By completing this project, we aim to determine the better approach to predicting economic patterns despite their chaotic nature.

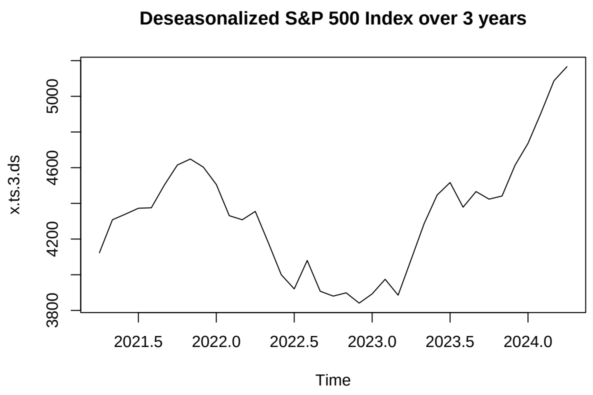
Preliminary EDA and Data Preparation

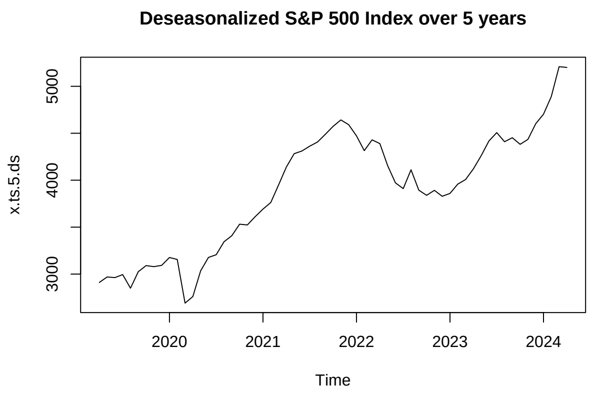
To simplify the frequentist (ARIMA) model creation, we take the monthly average of both the 3 year and the 5 year data sets instead of the daily value of the S&P 500.



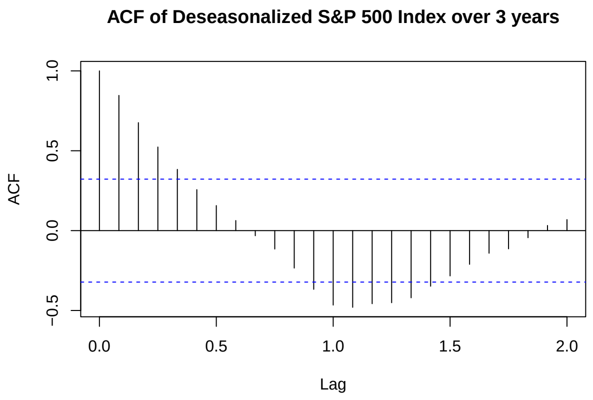


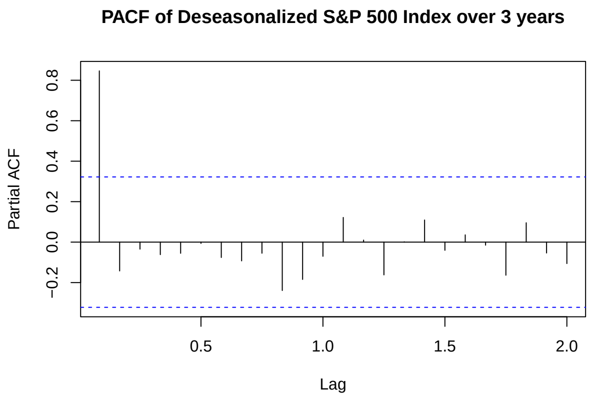
The time series plots shown for both 3 year and 5 year data sets demonstrate that the data is not stationary (mean is not constant), and some seasonality may be observed. As such, we remove seasonal variation from both datasets.

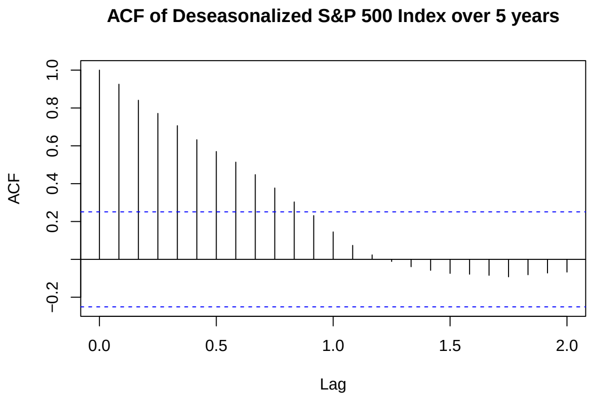


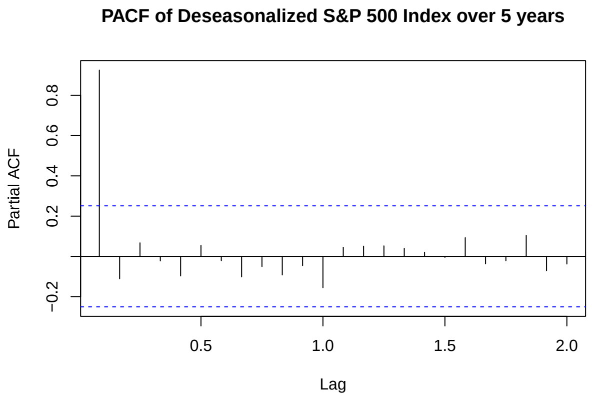


The autocorrelation functions (ACF) for both time series seem to have a sinusoidal pattern and tails off at around lag 1 (approximately 1 year) whereas their partial autocorrelation functions (PACF) seem to abruptly cut off after a lag of 1 month. As such, this suggests that the time series for both the 3 year and 5 year dataset could be represented as an AR(1) process









Model Development

We start by constructing a Bayesian model for the three year data, then we will make the model more complex. Acf and pacf analysis of the data showed that an $AR(1)$ process is suitable. Therefore, we incorporate this knowledge into our Bayesian model. Let i be the time index and y be the deaseasonalized time series.

$$\sigma \sim Exp\left(\frac{1}{100}\right)$$
$$y_i \sim N\left(y_{i-1}, \sigma\right)$$

```
data {
int N; // number of samples
real initial;
real final;
vector [N] y; // SP 500 index value
}
parameters {
  real <lower = 0> sigma;
}

model {
  sigma ~ exponential(1.0/100);
  for (i in 1:N){
    if (i == 1){
    y[i] ~ normal(initial, sigma);
} else{
  y[i] ~ normal(y[i-1], sigma);
}

}

}

generated quantities {
  vector [10] yhat;
  for (i in 1:10){
  if (i == 1){
  yhat[i] = normal_rng(final, sigma);
} else{
  yhat[i] =  normal_rng(yhat[i-1], sigma);

}

}

}
```

We use the data from April 2021 to June 2023 for training. Further, the data from July 2023 to April 2024 will be used for testing.

```
suppressMessages(require(rstan))
train = x.ts.3.ds[1:27]

fit = sampling(
  sp_model,
  seed = 1,
  chains = 1,
  data = list(
    N = length(train),
    initial = train[1],
    final = train[27],
    y = train
  ),
  iter = 10000
)
```

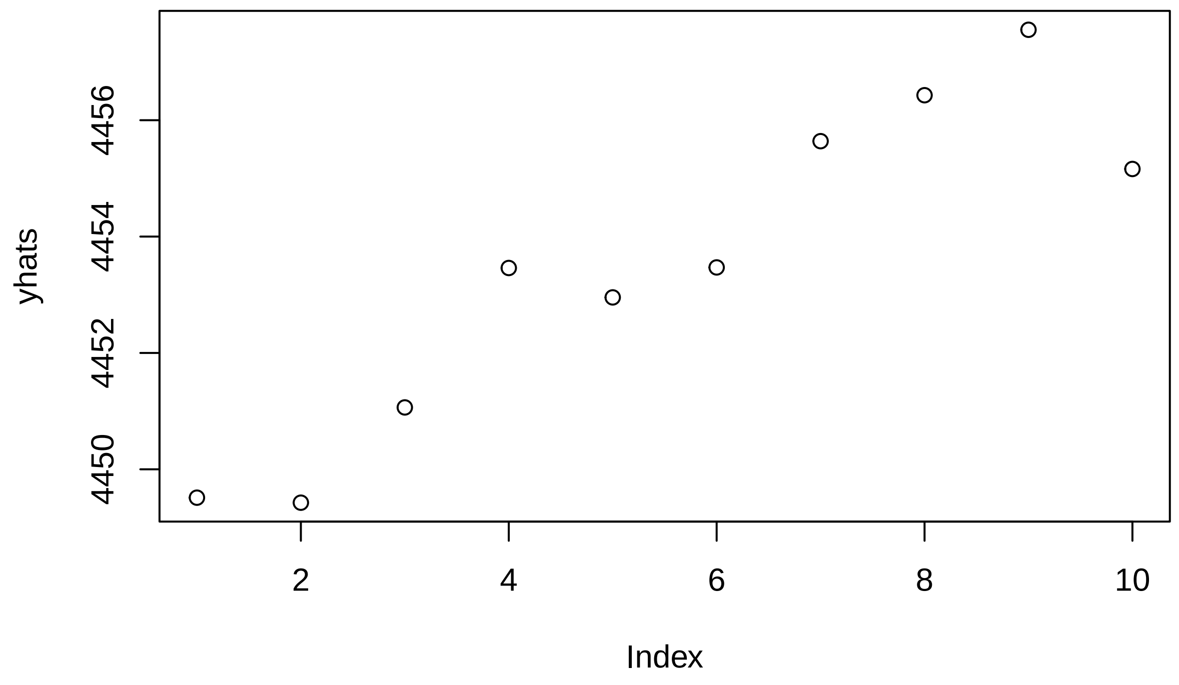
```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 2.1e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 1: Iteration: 3000 / 10000 [ 30%]  (Warmup)
## Chain 1: Iteration: 4000 / 10000 [ 40%]  (Warmup)
## Chain 1: Iteration: 5000 / 10000 [ 50%]  (Warmup)
## Chain 1: Iteration: 5001 / 10000 [ 50%]  (Sampling)
## Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.058 seconds (Warm-up)
## Chain 1:                0.058 seconds (Sampling)
## Chain 1:                0.116 seconds (Total)
## Chain 1:
```

```
samples = extract(fit)
fit
```

```
## Inference for Stan model: anon_model.
## 1 chains, each with iter=10000; warmup=5000; thin=1;
## post-warmup draws per chain=5000, total post-warmup draws=5000.
##
##             mean se_mean     sd    2.5%     25%     50%     75%   97.5% n_eff
## sigma     118.63    0.44  16.71   91.08  107.01  116.72  127.95  156.90  1461
## yhat[1]  4449.51    1.68 119.74 4214.07 4369.25 4449.64 4527.55 4683.70  5099
## yhat[2]  4449.43    2.41 169.89 4105.71 4337.36 4449.54 4559.68 4784.05  4961
## yhat[3]  4451.06    2.93 208.30 4032.73 4317.29 4450.48 4585.23 4869.97  5040
## yhat[4]  4453.46    3.43 243.13 3972.56 4293.72 4452.18 4614.07 4930.18  5032
## yhat[5]  4452.95    3.81 271.15 3922.74 4276.83 4454.10 4631.47 4971.21  5058
## yhat[6]  4453.47    4.15 294.81 3879.24 4255.08 4457.53 4647.27 5034.86  5050
## yhat[7]  4455.64    4.53 318.27 3835.46 4248.28 4460.64 4663.61 5072.48  4936
## yhat[8]  4456.43    4.80 338.26 3788.35 4236.19 4467.01 4672.36 5112.14  4972
## yhat[9]  4457.55    5.09 358.83 3740.82 4232.34 4460.12 4688.94 5159.62  4971
## yhat[10] 4455.16    5.36 376.97 3694.27 4212.14 4456.67 4702.69 5200.91  4944
## lp__      -138.75    0.02   0.73 -140.89 -138.92 -138.47 -138.30 -138.25  1670
##           Rhat
## sigma        1
## yhat[1]      1
## yhat[2]      1
## yhat[3]      1
## yhat[4]      1
## yhat[5]      1
## yhat[6]      1
## yhat[7]      1
## yhat[8]      1
## yhat[9]      1
## yhat[10]     1
## lp__         1
##
## Samples were drawn using NUTS(diag_e) at Mon Apr 15 07:03:49 2024.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```r
yhats = colMeans(samples$yhat)
```

```r
plot(yhats)
```


yhats
Index