Time Series Project

Liam Flaherty, Zhao Qu, and Zhijiang Yang

Professor Martin

NCSU: ST534-001

October 14, 2024

Contents

1 Introduction

In 2022, I had a small surgery on my back that kept me pretty immobile for months on end. Once I was able to start moving normally again, my strength and endurance were both fractions of what they were prior to undergoing the procedure.

To build back my health, I started running and lifting weights. Progress could be judged by direct measurement (e.g. how far I could run or how much weight I could lift), but hopefully, changes in these direct measurements would also manifest themselves in proxy measurements (e.g. an increase in strength would lead to an increase in muscle mass and thus body weight, while an increase in endurance would lead to better cardiovascular health).

To that end, I made it a goal to increase my body weight at a constraint of a steady resting heart rate. With a few exceptions, I recorded progress in these areas each morning from August 2022 to March 2023 by using a blood pressure and heart rate gauge I bought from CVS and a bathroom scale. We reserve the data post February 2023 as the test set, and plot the training set in Figures 1.1-1.2 below.

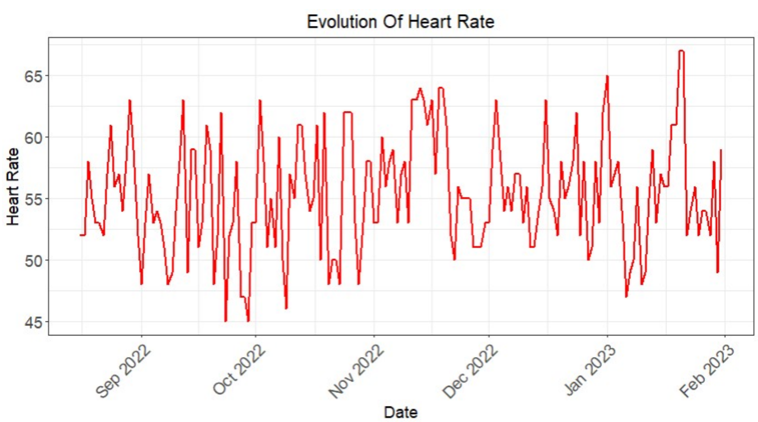

Figure 1.1: Evolution Of Heart Rate

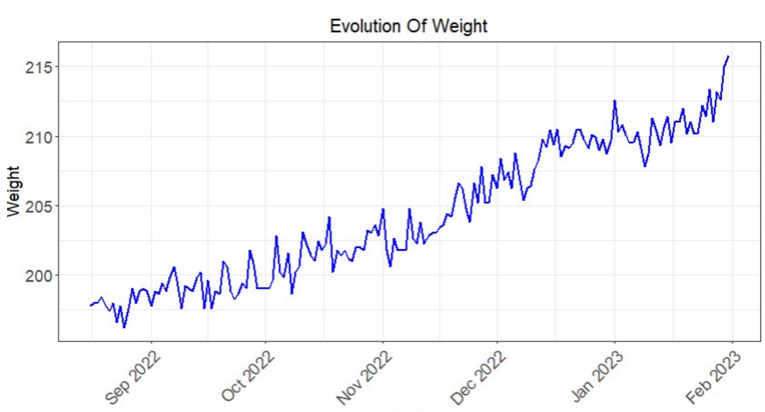

Figure 1.2: Evolution Of Weight

2 Model Selection

2.1 White Noise Test

It is clear from inspection that the time series for weight is not white noise. We use the Ljung-Box Q Test to determine whether we need to fit a model for the heart rate data. The function `Box.test()` in R provides a nice way to look at this. Upon running this function on our data, we get the below output in Figure 2.1. With a p-value of about 0.004, we reject the null hypothesis of "white noise" under a significance level of $\alpha = 0.05$.

```
> whitenoise6=Box.test(ts_hr,                #Do we need to fit model?#
+                      lag=6,
+                      type="Ljung-Box")
> whitenoise6                                #p small \implies yes#

        Box-Ljung test

data:  ts_hr
X-squared = 19.069, df = 6, p-value = 0.004048
```

Figure 2.1: R Output For Ljung-Box Q Test

2.2 (Weak) Stationarity Test

We can test whether or not we need to take a difference in either dataset in order to make the data stationary (test if there is a unit root) with the Augmented Dickey-Fuller Test. At least visually, it appears there is trending in the weight dataset, but constant mean in the heart rate dataset. This is borne out by a formal test of the data, which we show in Figures 2.2-2.3 below. Note that there is not enough evidence to reject the null hypothesis of "non-stationary" for the weight data under a significance level of $\alpha = 0.05$.

```
> adf_result_weight=suppressWarnings(adf.test(ts_weight))
> adf_result_weight                              #difference needed#

        Augmented Dickey-Fuller Test

data:  ts_weight
Dickey-Fuller = -2.8013, Lag order = 5, p-value = 0.2418
alternative hypothesis: stationary
```

Figure 2.2: ADF Output For Weight

```
> adf_result_hr=suppressWarnings(adf.test(ts_hr))
> adf_result_hr

        Augmented Dickey-Fuller Test

data:  ts_hr
Dickey-Fuller = -4.775, Lag order = 5, p-value = 0.01
alternative hypothesis: stationary
```

Figure 2.3: ADF Output For Heart Rate

Indeed, after taking a difference (Figure 2.4), we see the time series appears significantly more stationary than previously (Figure 1.2).

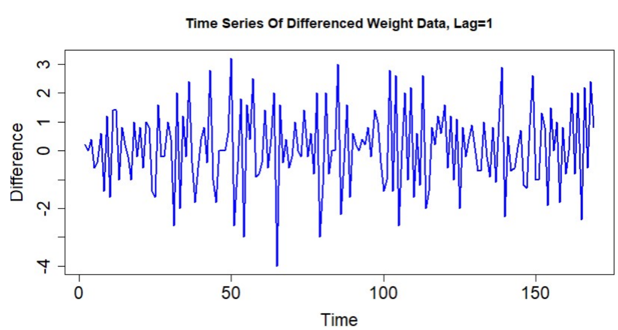

Figure 2.4: Time Series After Difference

2.3 Autocorrelation And Partial Autocorrelation Functions

The autocorrelation and partial autocorrelation for our heart rate data (Figure 2.5) and weight data (Figure 2.6) are shown below.

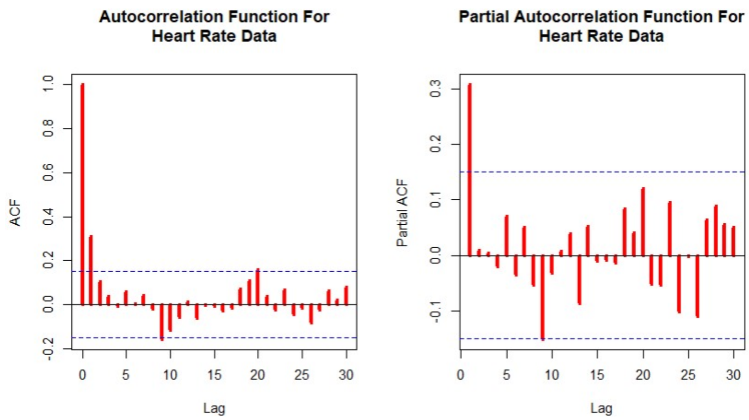

Figure 2.5: ACF And PACF For Heart Rate

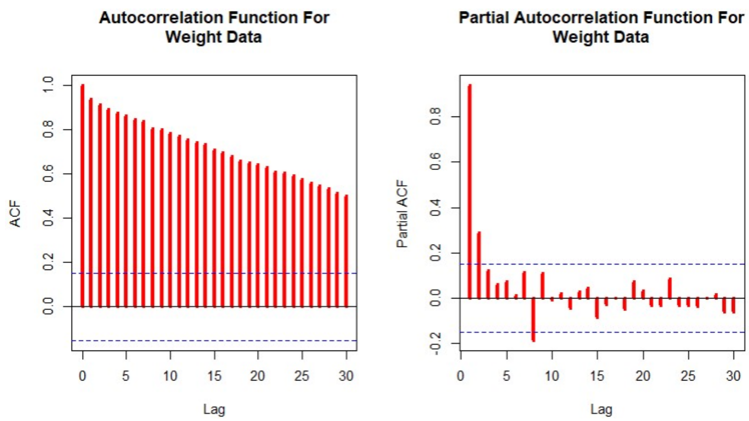

Figure 2.6: ACF And PACF For Weight

As expected, the ACF for the weight data refuses to die out; the data is heavily correlated. After taking a difference, we see the P/ACF plots in Figure 2.7 below.

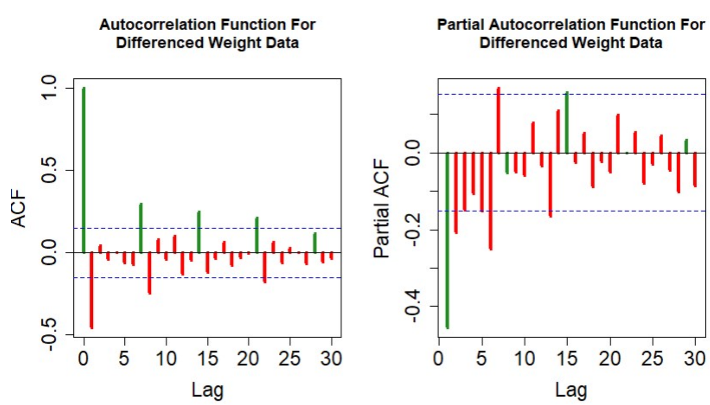

Figure 2.7: ACF And PACF For Differenced Data

2.4 Seasonality

Notice that the ACF for the differenced data in Figure 2.7 above has spikes at lags of 7, 14, and 21 (the seasonal lags are highlighted in green). This indicates that we might try fitting a seasonal component to the data. There is a physical explanation for this seasonality as well– I didn't run on Sunday's and often ate out on the weekend.

To account for this seasonality, we can try to fit a SARIMA with $s = 7$. Since the spikes in the ACF are not growing, we may not need to take a seasonal difference. Nevertheless, we try taking one and see how the P/ACF plots look. They are shown in Figure 2.8 below.

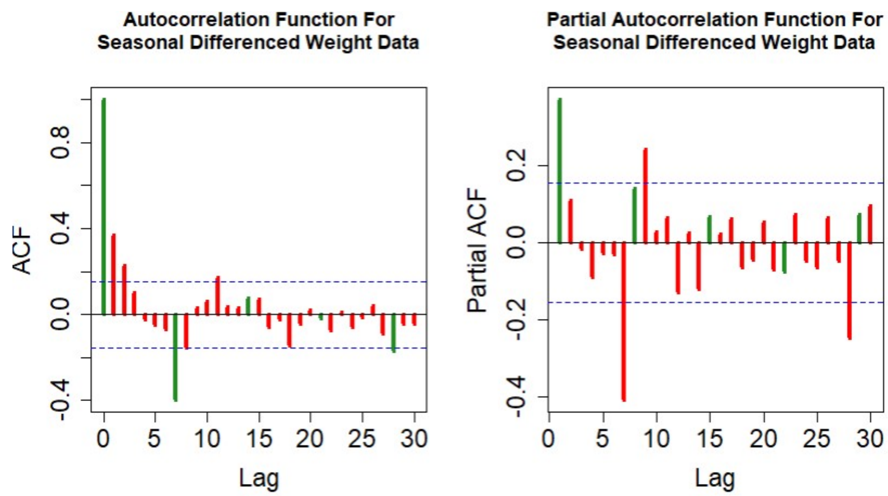

Figure 2.8: ACF And PACF Of Seasonally Differenced Weight Data

There are still some large spikes in the PACF well out into the data, so taking just a seasonal difference may not be sufficient. When taking both a seasonal difference ($D = 1$) and then a regular difference ($d = 1$), we get the plots for our P/ACF in Figure 2.9.

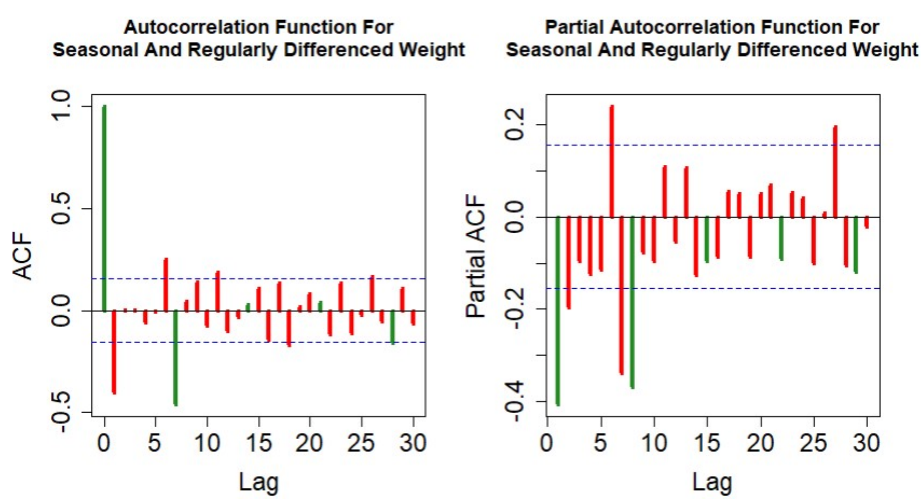

Figure 2.9: ACF And PACF Of Seasonally And Regularly Differenced Weight

2.5 Suggested Models

From Subsection 2.4, there is no clear answer as to which differencing combination (the d and D terms) is best to deal with the weight data.

Based on the ACF and PACF from the regularly differenced data ($d = 1$) from Figure 2.7, we see the seasonal lags of the ACF gradually die out, while the seasonal lags of the PACF have a spike at lag 2 (so, an ARIMA(2,0,0) or ARIMA(2,0,3) for the seasonal component might make sense). The regular lags in the ACF do not completely die out with spikes well out into the series, though the last large spike is at lag 1. The regular lags in the PACF gradually die out, despite having a large spike at lag 6 (so, an ARIMA(0,1,1) for the regular component might make sense). This first bit of our recommended models are ARIMA(0,1,1)(2,0,0)7 and ARIMA(0,1,1)(2,0,3)7.

Based on the ACF and PACF from the seasonally differenced data ($D = 1$) from Figure 2.8, we see the seasonal lags in the ACF die out after a large first spike, while the seasonal lags of the PACF immediately dissipate (so, an ARIMA(0,1,1) or ARIMA(0,1,0) for the seasonal component might make sense). The regular lags in the ACF have meaningful spikes at lags 1 and 2 before having a sinusoidal decay. The regular lags in the PACF do not really die off (so, an ARIMA(0,0,1) or ARIMA(0,0,2) or ARIMA(0,0,3) for the regular component might make sense). The second bit of our recommended models are ARIMA(0,0,1)(0,1,1)7, ARIMA(0,0,1)(0,1,0)7, ARIMA(0,0,2)(0,1,1)7, and ARIMA(0,0,2)(0,1,0)7.

Based on the ACF and PACF from the data that is both seasonally differenced and regularly differenced ($d = 1, D = 1$), we see one large spike in the seasonal lag of the ACF and one large spike in the seasonal lag of the PACF (so, an ARIMA(1,1,0) or ARIMA(0,1,1) or ARIMA(1,1,1) for the seasonal component might make sense). The regular lags in the ACF and PACF do not really die off, but maybe an ARIMA(1,1,0) or ARIMA(0,1,1) or ARIMA(1,1,1) could work. The third and final bit of our recommended models are ARIMA(1,1,1)(1,1,0)7, ARIMA(1,1,1)(0,1,1)7, ARIMA(1,1,1)(1,1,1)7, ARIMA(0,1,1)(1,1,0)7, ARIMA(0,1,1)(0,1,1)7, ARIMA(0,1,1)(1,1,1)7, ARIMA(1,1,0)(1,1,0)7, ARIMA(1,1,0)(0,1,1)7, and ARIMA(1,1,0)(1,1,1)7.

The heart rate data is much simpler. Based on the ACF and PACF for the Heart Rate Data in Figure 2.5, we suggest an AR(1). This is because the last large spike in the ACF and PACF are both at lag one, and the ACF does not completely die off.

2.6 Model Diagnostics

We prioritize the models we identified in subsection 2.5, but we also have lots of computational power to try many different models.

We utilize this power by trying all seasonal ARIMA models with $p, q, P,$ and Q terms less than 5 and d and D terms less than 2. For each of the $5^4 \times 2 \times 2 = 2500$ models, we compute the AIC and BIC for model evaluation, and the p-value from the Ljung-Box Q test to see if the residuals from our model are actually white noise. The top 15 models in terms of BIC are shown in Figure 2.10 below.

```
> df_weight[1:15,]
```

	ARIMAs_model	aic	bic	LBtest
307	ARIMA(0,1,1)(0,1,1)7	509.46	518.71	0.73
807	ARIMA(1,1,1)(0,1,1)7	509.92	522.25	0.81
357	ARIMA(0,1,2)(0,1,1)7	510.49	522.82	0.79
317	ARIMA(0,1,1)(1,1,1)7	511.35	523.68	0.66
308	ARIMA(0,1,1)(0,1,2)7	511.39	523.72	0.69
1307	ARIMA(2,1,1)(0,1,1)7	509.31	524.71	0.88
407	ARIMA(0,1,3)(0,1,1)7	509.43	524.84	0.85
857	ARIMA(1,1,2)(0,1,1)7	510.31	525.72	0.86
327	ARIMA(0,1,1)(2,1,1)7	510.54	525.95	0.67
557	ARIMA(1,0,1)(0,1,1)7	513.60	525.95	0.71
309	ARIMA(0,1,1)(0,1,3)7	510.74	526.14	0.65
817	ARIMA(1,1,1)(1,1,1)7	511.52	526.93	0.74
808	ARIMA(1,1,1)(0,1,2)7	511.66	527.07	0.77
367	ARIMA(0,1,2)(1,1,1)7	512.24	527.65	0.72
358	ARIMA(0,1,2)(0,1,2)7	512.33	527.73	0.75

Figure 2.10: ARIMA Model Diagnostics

See that our recommended ARIMA(0,1,1)(0,1,1) had the best BIC. Also notice that all the top models had both a seasonal and regular difference. We finally note that in terms of AIC, the ARIMA(0,1,1)(0,1,1) was also a top performer, and only models with much more terms (e.g. ARIMA(2,1,1)(1,1,4)7), bested it by that metric. Since we are using the models for prediction, we prefer parsimony and so base our decision on the metric that is less forgiving to added parameters; our model choice is the ARIMA(0,1,1)(0,1,1)7.

The heart rate data was stationary to begin with; we only need to consider ARMA models. The top model in terms of BIC (of all combinations of p and q less than 5) was our suggested AR(1). The top ten models are shown in Figure 2.11 below.

```
> df_hr[1:10,]
```

	ARIMAs_model	aic	bic	LBtest
7	ARMA(1,0)	993.33	1002.72	0.86
2	ARMA(0,1)	994.81	1004.20	0.78
8	ARMA(1,1)	995.32	1007.84	0.86
13	ARMA(2,0)	995.32	1007.84	0.86
3	ARMA(0,2)	995.58	1008.10	0.85
4	ARMA(0,3)	997.12	1012.76	0.87
9	ARMA(1,2)	997.32	1012.97	0.86
19	ARMA(3,0)	997.32	1012.97	0.86
14	ARMA(2,1)	997.33	1012.98	0.86
1	ARMA(0,0)	1008.24	1014.50	0.02

Figure 2.11: ARIMA Model Diagnostics

3 Parameter Selection

3.1 Heart Rate Data

We fit both series with two models each using the `forecast` package from R. In choosing the coefficients, we are selecting those parameter values which minimize the conditional least squares.

For the heart rate data, the top performing model was our suggested AR(1). Our model is (where $a_t \sim N(0, 20.16)$):

$$a_t = \pi(B)\widetilde{Z}_t \tag{3.1}$$
$$a_t \approx (1 - 0.3086B)(Z_t - 55.3846) \tag{3.2}$$
$$Z_t \approx 55.3846 + 0.3086(Z_{t-1} - 55.3846) + a_t \tag{3.3}$$

```
> hr_ar1

Call:
arima(x = ts_hr, order = c(1, 0, 0), method = "ML")

Coefficients:
         ar1  intercept
      0.3086    57.3851
s.e.  0.0731     0.4983

sigma^2 estimated as 20.16:  log likelihood = -493.67,  aic = 993.33
```

Figure 3.1: R Code For Heart Rate Model Coefficients

The second best model was an MA(1). The model fit is (where $a_t \sim N(0, 20.34)$):

$$\widetilde{Z}_t = \psi(B)a_t \tag{3.4}$$
$$(Z_t - 55.3846) \approx (1 + 0.2283B)a_t \tag{3.5}$$
$$Z_t \approx 55.3846 + a_t + 0.2283a_{t-1} \tag{3.6}$$

```
> hr_ma1

Call:
arima(x = ts_hr, order = c(0, 0, 1), method = "ML")

Coefficients:
         ma1  intercept
      0.2883    57.3906
s.e.  0.0704     0.4463

sigma^2 estimated as 20.34:  log likelihood = -494.4,  aic = 994.81
```

Figure 3.2: R Code For Heart Rate Model Coefficients

3.2 Weight Data

We give the coefficients to the Seasonal ARIMA models that we fit for the weight data below. The best model in terms of BIC was our suggested SARIMA(0,1,1)(0,1,1). Our model is (where $a_t \sim N(0, 1.268)$):

$$\pi(B)\Pi(B^s)(1-B)^d(1-B^s)^D Z_t = \psi(B)\Psi(B^s)a_t \tag{3.7}$$
$$(1-B)(1-B^7)Z_t \approx (1-0.6659B)(1-0.8147B^7)a_t \tag{3.8}$$
$$Zt \approx Z_{t-1} + Z_{t-7} - Z_{t-8} + a_t - 0.6659a_{t-1} - 0.8147a_{t-7} + 0.5425a_{t-8} \tag{3.9}$$

```
> weight_sarima011011

Call:
arima(x = ts_weight, order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1),
    period = 7), method = "ML")

Coefficients:
          ma1     sma1
      -0.6659  -0.8147
s.e.   0.0826   0.0886

sigma^2 estimated as 1.268:  log likelihood = -251.73,  aic = 509.46
```

Figure 3.3: R Code For Weight Data Model Coefficients

The best model in terms of AIC was a SARIMA(2,1,1)(1,1,4). Our model is (where $a_t \sim N(0, 1.085)$):

$$\pi(B)\Pi(B^s)(1-B)^d(1-B^s)^D Z_t = \psi(B)\Psi(B^s)a_t \tag{3.10}$$
$$(1+0.2680B+0.2111B^2)(1-0.7620B^7)(1-B)(1-B^7)Z_t \approx \tag{3.11}$$
$$(1-0.9142B)(1-0.0006B^7-0.6255B^{14}-0.1021B^{21}-0.2686B^{28})a_t \tag{3.12}$$

```
> weight_sarima211114

Call:
arima(x = ts_weight, order = c(2, 1, 1), seasonal = list(order = c(1, 1, 4),
    period = 7), method = "ML")

Coefficients:
         ar1     ar2      ma1     sar1     sma1     sma2     sma3     sma4
      0.2680  0.2111  -0.9142  -0.7620  -0.0006  -0.6255  -0.1021  -0.2686
s.e.  0.1071  0.0985   0.0740   0.1412   0.4828   0.4731   0.1948   0.1551

sigma^2 estimated as 1.085:  log likelihood = -245.62,  aic = 509.25
```

Figure 3.4: R Code For Weight Data Model Coefficients

4 Forecasting

With our top models in hand, we try to forecast our series. We forecast our series out a month, and compare it to our test data that we reserved from the outset. The forecast can be done automatically with R using the `forecast()` function.

The results for the weight data are shown in Figure 4.1 below. The red shading refers to the 95% Prediction Interval for the SARIMA(2,1,1)(1,1,4)7 model while the blue shading refers to the 95% Prediction Interval for the SARIMA(0,1,1)(0,1,1)7 model.

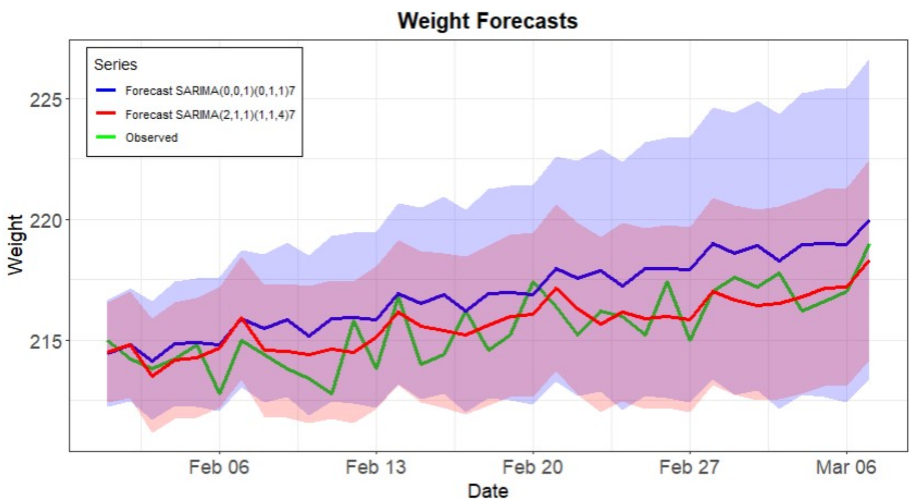

Figure 4.1: Comparison Of Predicted And Observed Values For Weight Data

While an argument could have been made that the more comprehensive model favored by AIC overfits to the training data, it actually does a better job at forecasting our test data compared to the more parsimonious model we suggested (out of sample RMSE of 0.95 compared to out of sample RMSE of 1.73). In either case, see how the prediction bounds grow the larger we move from observed data. This is only natural– our uncertainty about the future grows based on the time.

Since the recommended model for the heart rate data was an AR(1), the forecast will be mean-reverting; the second term in out model $Z_t \approx 55.3846 + 0.3086(Z_{t-1} - 55.3846) + a_t$ becomes smaller and smaller. Our forecasts are:

$$\widehat{Z_{169}}(1) = 55.38462 + 0.3086(59 - 55.38462) \approx 56.500$$
$$\widehat{Z_{169}}(2) = 55.38462 + 0.3086(56.500 - 55.38462) \approx 55.729$$
$$\widehat{Z_{169}}(3) = 55.38462 + 0.3086(55.729 - 55.38462) \approx 55.491$$
$$\vdots$$

Indeed, after twelve units, our predictions stabilize up to the fifth decimal.

```
> forecast_hr_ar1
         date forecast    lower    upper observed      resid
1  2023-02-01 56.50081 47.70026 65.30136       58  1.4991863
2  2023-02-02 55.72947 46.51929 64.93965       55 -0.7294680
3  2023-02-03 55.49140 46.24315 64.73965       55 -0.4914009
4  2023-02-04 55.41792 46.16605 64.66979       50 -5.4179241
5  2023-02-05 55.39525 46.14303 64.64746       53 -2.3952464
6  2023-02-06 55.38825 46.13600 64.64049       54 -1.3882471
7  2023-02-07 55.38609 46.13384 64.63834       54 -1.3860869
8  2023-02-08 55.38542 46.13317 64.63767       59  3.6145799
9  2023-02-09 55.38521 46.13296 64.63746       59  3.6147856
10 2023-02-10 55.38515 46.13290 64.63740       61  5.6148491
11 2023-02-11 55.38513 46.13288 64.63738       55 -0.3851312
12 2023-02-12 55.38513 46.13287 64.63738       55 -0.3851252
```

5 Appendix

```
##########Code Written By Liam Flaherty For ST534 Final Project##########
#####1. Initial Data######
###1a. Load in data and required packages###
library(tidyverse)
library(scales)
library(forecast)

path="C:/Users/LiamFlaherty/Documents/Academics/ST534 Time Series/Project/weight.csv"
weight=read.csv(path)

weight=weight |>
  mutate(Date=as.Date(Date)) |>
  mutate(Day=weekdays(Date)) |>
  select(Date, Day, Weight, Lower, Upper, HR)

str(weight)
summary(weight)

###1b. Split into training and test###
train=weight[which(weight$Date<"2023-02-01"),]
test=weight[which(weight$Date>="2023-02-01"),]

#####2. Exploratory Data Analysis#####
###2a. Heart Rate###
ggplot(train, aes(x=Date, y=HR)) +
  geom_line(color="red", linewidth=1) +
  labs(title="Evolution of Heart Rate",
       x="Date",
       y="Heart Rate") +
  scale_x_date(
    date_breaks="1 month",
    date_labels="%b %Y") +
  theme_bw() +
  theme(
    plot.title=element_text(hjust=0.5, size=16),  #Center the title#
    axis.text=element_text(size=14),
    axis.title=element_text(size=14),
    axis.text.x=element_text(angle=45, hjust=1))

hist(train$HR,
     main="Histogram of HR (8/2022 - 3/2023)",
     xlab="Heart Rate",
     ylab="Frequency",
     xlim=c(45,70),
     col="Red")

sd(train$HR)

###2b. Weight###
ggplot(train, aes(x=Date, y=Weight)) +
  geom_line(color="blue", linewidth=1) +
  labs(title="Evolution of Weight",
       x="Date",
       y="Weight") +
  scale_x_date(
    date_breaks="1 month",
    date_labels="%b %Y") +
  theme_bw() +
  theme(
    plot.title=element_text(hjust=0.5, size=16),  #Center the title#
    axis.text=element_text(size=14),
    axis.title=element_text(size=14),
    axis.text.x=element_text(angle=45, hjust=1))

#####3. Analysis#####
###3a. Convert to time series###
ts_weight=ts(train$Weight)              #convert to time series object#
ts_hr=ts(train$HR)

###3b. White Noise Test###
#Clear that weight is not white noise#
whitenoise6=Box.test(ts_hr,                #Do we need to fit model?#
                     lag=6,
                     type="Ljung-Box")
whitenoise6                                #p small \implies yes#

whitenoise12=Box.test(ts_hr,               #Do we need to fit model?#
                      lag=12,
                      type="Ljung-Box")
whitenoise12                               #p small \implies yes#

###3c. Test for stationarity###
adf_result_weight=suppressWarnings(adf.test(ts_weight))
adf_result_weight                          #difference needed#
```

```r
adf_result_hr=suppressWarnings(adf.test(ts_hr))
adf_result_hr                                           #stationary#



###3c. Initial P/ACF For HR###
par(mfrow=c(1,2))                   #split the display to show two figures in one plot#

acf(ts_hr,
    main=paste0("Autocorrelation Function For", "\n", "Heart Rate Data"),
    lag.max=30,
    ci.col="blue",
    col="red",
    lwd=4)

pacf(ts_hr,
     main=paste0("Partial Autocorrelation Function For", "\n", "Heart Rate Data"),
     lag.max=30,
     ci.col="blue",
     col="red",
     lwd=4)

par(mfrow=c(1,1))                    #back to one figure per plot#



###3d. Initial P/ACF For Weight###
par(mfrow=c(1,2))

acf(ts_weight,
    main=paste0("Autocorrelation Function For", "\n", "Weight Data"),
    lag.max=30,
    ci.col="blue",
    col="red",
    lwd=4)

pacf(ts_weight,
     main=paste0("Partial Autocorrelation Function For", "\n", "Weight Data"),
     lag.max=30,
     ci.col="blue",
     col="red",
     lwd=4)

par(mfrow=c(1,1))



###3e. P/ACF For Weight With Regular Difference###
train_diff=diff(ts_weight, lag=1)

par(cex.axis=1.5, cex.lab=1.5)
plot(train_diff,
     main="Time Series Of Differenced Weight Data, Lag=1",
     xlab="Time",
     ylab="Difference",
     cex.axis=2,
     cex.lab=2,
     col="blue",
     lwd=2)

par(mfrow=c(1,2))
acf(train_diff,
    main=paste0("Autocorrelation Function For", "\n", "Differenced Weight Data"),
    lag.max=30,
    ci.col="blue",
    col=ifelse((0:30 %% 7)==0, "forestgreen", "red"),
    lwd=4)

pacf(train_diff,
     main=paste0("Partial Autocorrelation Function For", "\n", "Differenced Weight Data"),
     lag.max=30,
     ci.col="blue",
     col=ifelse((0:30 %% 7)==0, "forestgreen", "red"),
     lwd=4)

par(mfrow=c(1,1))



###3f. P/ACF For Weight With Just Seasonal Difference###
ts_weight_sdiff=diff(ts_weight, lag=7)                  #just D=1#

par(mfrow=c(1,2))

acf(ts_weight_sdiff,
    main=paste0("Autocorrelation Function For", "\n", "Seasonal Differenced Weight Data"),
    lag.max=30,
    ci.col="blue",
    col=ifelse((0:30 %% 7)==0, "forestgreen", "red"),
    lwd=4)

pacf(ts_weight_sdiff,
     main=paste0("Partial Autocorrelation Function For", "\n", "Seasonal Differenced Weight Data"),
     lag.max=30,
     ci.col="blue",
     col=ifelse((0:30 %% 7)==0, "forestgreen", "red"),
     lwd=4)

par(mfrow=c(1,1))
```

```
###3g. P/ACF For Weight With Both Differences###
ts_weight_both=diff(diff(ts_weight, lag=7), lag=1)                #D=1, d=1#

par(mfrow=c(1,2))

acf(ts_weight_both,
    main=paste0("Autocorrelation Function For", "\n", "Seasonal And Regularly Differenced Weight"),
    lag.max=30,
    ci.col="blue",
    col=ifelse((0:30 %% 7)==0, "forestgreen", "red"),
    lwd=4)

pacf(ts_weight_both,
     main=paste0("Partial Autocorrelation Function For", "\n", "Seasonal And Regularly Differenced Weight"),
     lag.max=30,
     ci.col="blue",
     col=ifelse((0:30 %% 7)==0, "forestgreen", "red"),
     lwd=4)

par(mfrow=c(1,1))




#####4. Try A Bunch Of Models#####
###4a. For Weight Data###
ARIMAs_model_weight=vector()
aic_weight=vector()
bic_weight=vector()
LBtest_weight=vector()
s=7                        #From Analysis#
m=5                        #the number of MA and AR terms to try#
i=0                        #to keep track of iterations

for (p in 1:m) {
  for (d in 1:2) {
    for (q in 1:m) {
      for (P in 1:m) {
        for (D in 1:2) {
          for (Q in 1:m) {
            i=i+1
            print(paste0("i=", round(i/(m^4*2*2), 2) ))     #where we're at in the process#

            mymodel=paste0("ARIMA(", p-1, ",", d-1, ",", q-1, ")(", P-1, ",", D-1, ",", Q-1, ")",s)

            setTimeLimit(cpu=3, elapsed=3)                   #otherwise would take forever#
            model_result=tryCatch({                          #for convergence problems#
              model=arima(ts_weight,
                          order=c(p-1,d-1,q-1),
                          seasonal=list(order=c(P-1,D-1,Q-1), period=s),
                          method="ML")                       #By Maximum Likelihood#

              ARIMAs_model_weight[i]=mymodel
              aic_weight[i]=round(AIC(model),2)
              bic_weight[i]=round(BIC(model),2)
              LBtest_weight[i]=round(Box.test(residuals(model), lag=21, type="Ljung-Box")$p.value,2)

            }, error=function(e) {                           #for convergence problems#
              ARIMAs_model_weight[i]=mymodel
              aic_weight[i]=999
              bic_weight[i]=999
              LBtest_weight[i]=999
            })
            setTimeLimit(cpu=Inf, elapsed=Inf)

          }
        }
      }
    }
  }
}

df_weight=data.frame(ARIMAs_model=ARIMAs_model_weight,
                     aic=aic_weight,
                     bic=bic_weight,
                     LBtest=LBtest_weight)
df_weight=df_weight[which(df_weight$bic<999),]
df_weight=df_weight[order(df_weight$bic),]
df_weight[1:15,]

save(df_weight, file="modelfit_weight.R")
load("modelfit_weight.R")                                    #so don't have to run this part of the code#



###4b. For HR Data###
ARIMAs_model_hr=vector()
aic_hr=vector()
bic_hr=vector()
LBtest_hr=vector()
m=5                        #the number of MA and AR terms to try#
i=0

for (p in 0:m) {
  for (q in 0:m) {
    i=i+1
    mymodel=paste0("ARMA(", p, ",", q, ")")
```

```
    model=arima(ts_hr,
                order=c(p,0,q),
                method="ML")              #by Maximum Likelihood#

    ARIMAs_model_hr[i]=mymodel
    aic_hr[i]=round(AIC(model),2)
    bic_hr[i]=round(BIC(model),2)
    LBtest_hr[i]=round(Box.test(residuals(model), lag=21, type="Ljung-Box")$p.value,2)
  }
}

df_hr=data.frame(
  ARMA_model=ARIMAs_model_hr,
  aic=aic_hr,
  bic=bic_hr,
  LBTest=LBtest_hr)
df_hr=df_hr[order(df_hr$bic),]
df_hr[1:10,]

save(df_hr, file="modelfit_hr.R")
load("modelfit_hr.R")                              #so don't have to run this part of the code#



#####4c. Getting parameter weights###
hr_ar1=arima(ts_hr,                               #best in terms of BIC#
             order=c(1,0,0),
             method="ML")

hr_ma1=arima(ts_hr,                               #our recommendation; 2nd best in AIC and BIC#
             order=c(0,0,1),
             method="ML")

weight_sarima211114=arima(ts_weight,              #best in terms of AIC#
                          order=c(2,1,1),
                          seasonal=list(order=c(1,1,4), period=7),
                          method="ML")

weight_sarima011011=arima(ts_weight,              #best in terms of BIC; our recommendation#
                          order=c(0,1,1),
                          seasonal=list(order=c(0,1,1), period=7),
                          method="ML")

hr_ar1
hr_ma1
weight_sarima211114
weight_sarima011011



#####5. Forecast#####
###5a. Weight Data###
forecast_weight_aic=predict(weight_sarima211114,
                            n.ahead=nrow(test))

forecast_weight_aic=data.frame(
  date=seq(from=test$Date[1], to=test$Date[nrow(test)], by="day"),
  forecast=forecast_weight_aic$pred,
  lower=forecast_weight_aic$pred-1.96*forecast_weight_aic$se,
  upper=forecast_weight_aic$pred+1.96*forecast_weight_aic$se,
  observed=test$Weight,
  resid=test$Weight-forecast_weight_aic$pred
)

forecast_weight_bic=predict(weight_sarima011011,
                            n.ahead=nrow(test))

forecast_weight_bic=data.frame(
  date=seq(from=test$Date[1], to=test$Date[nrow(test)], by="day"),
  forecast=forecast_weight_bic$pred,
  lower=forecast_weight_bic$pred-1.96*forecast_weight_bic$se,
  upper=forecast_weight_bic$pred+1.96*forecast_weight_bic$se,
  observed=test$Weight,
  resid=test$Weight-forecast_weight_bic$pred
)

rmse_weight_aic=(sum(forecast_weight_aic$resid^2)/nrow(forecast_weight_aic))^(0.5)
rmse_weight_bic=(sum(forecast_weight_bic$resid^2)/nrow(forecast_weight_aic))^(0.5)
rmse_weight_aic                                    #just curious#
rmse_weight_bic                                    #just curious#

forecast_weight_aic
forecast_weight_bic



###5b. Plot Weight Data###
ggplot() +
  geom_line(data=forecast_weight_bic,
            aes(x=date, y=observed, color="Observed"),
            size=1.2) +
  geom_line(data=forecast_weight_bic,
            aes(x=date, y=forecast, color="Forecast SARIMA(0,0,1)(0,1,1)7"),
            size=1.2) +
  geom_ribbon(data=forecast_weight_bic,
              aes(x=date, ymin=lower, ymax=upper),
              fill="blue",
              alpha=0.2) +
  geom_line(data=forecast_weight_aic,
```