Econ 144 Project 3

Nicholas Cassol-Pawson

June 7, 2024

Contents


```
library(fpp3)
library(forecast)
library(fable.prophet)
library(tidyverse)
library(tseries)
library(vars)
library(rugarch)
```

I. Introduction

```
data <- read.csv("BWIData.csv") %>%
  transmute("Year" = year(as.Date(DATE)), "Month" = month(as.Date(DATE)),
            "Day" = day(as.Date(DATE)), "Precip" = PRCP) %>%
  filter(Year != 2024) %>%
  group_by(Year, Month) %>%
  summarise("PrecipMean" = mean(Precip))
```

```
## `summarise()` has grouped output by 'Year'. You can override using the
## `.groups` argument.
```

```
precip <- ts(data$PrecipMean, freq = 12, start = 1980)
```

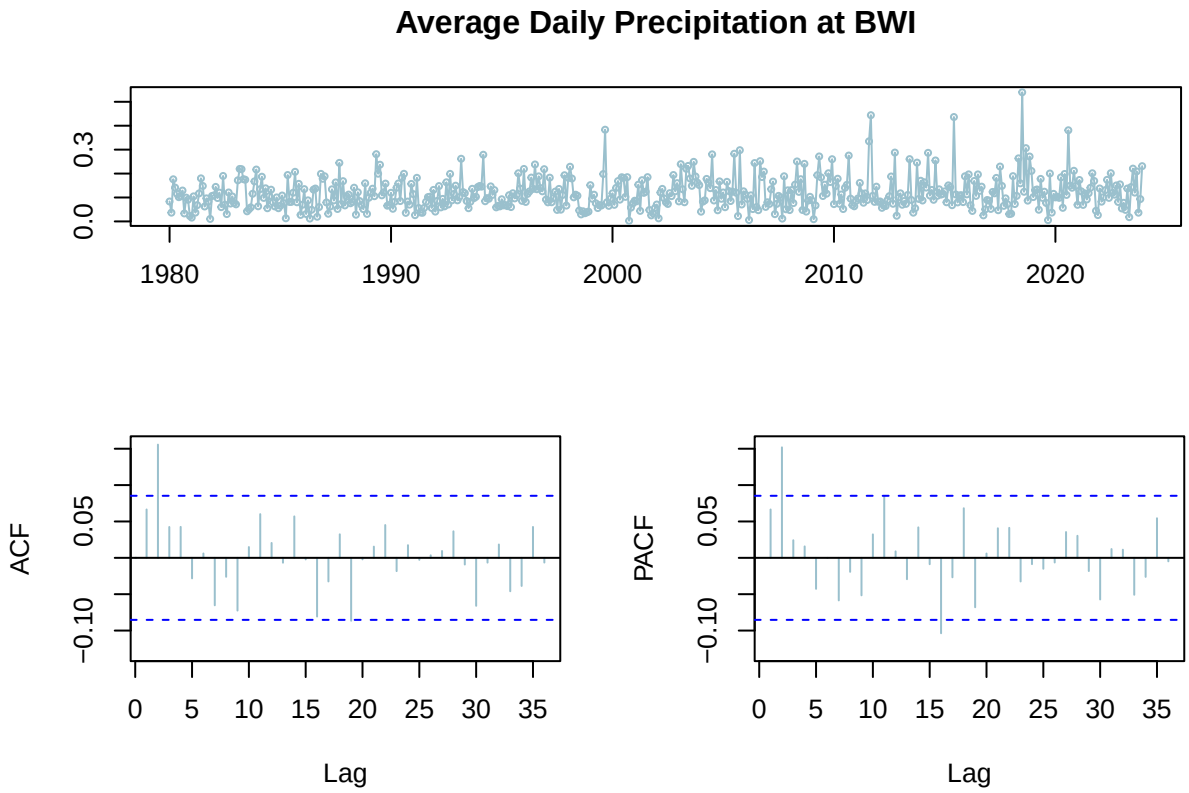
Washington, DC, is located at the center of the East coast, so it has a typical Southern climate with large amounts of rainfall in the summer and a New England-style one with years of snow in the winter. However, precipitation amounts in the city vary year to year, so it would be interesting to predict future rainfall amounts from current ones. We have the each month's average daily amount of precipitation in inches, observed at the NOAA weather data collection station at Baltimore/Washington International Airport.

II. Results

Data Exploration

We will eventually try to forecast the data using several different models, but we begin by exploring the data. We first plot the precipitation series, its ACF and its PACF.

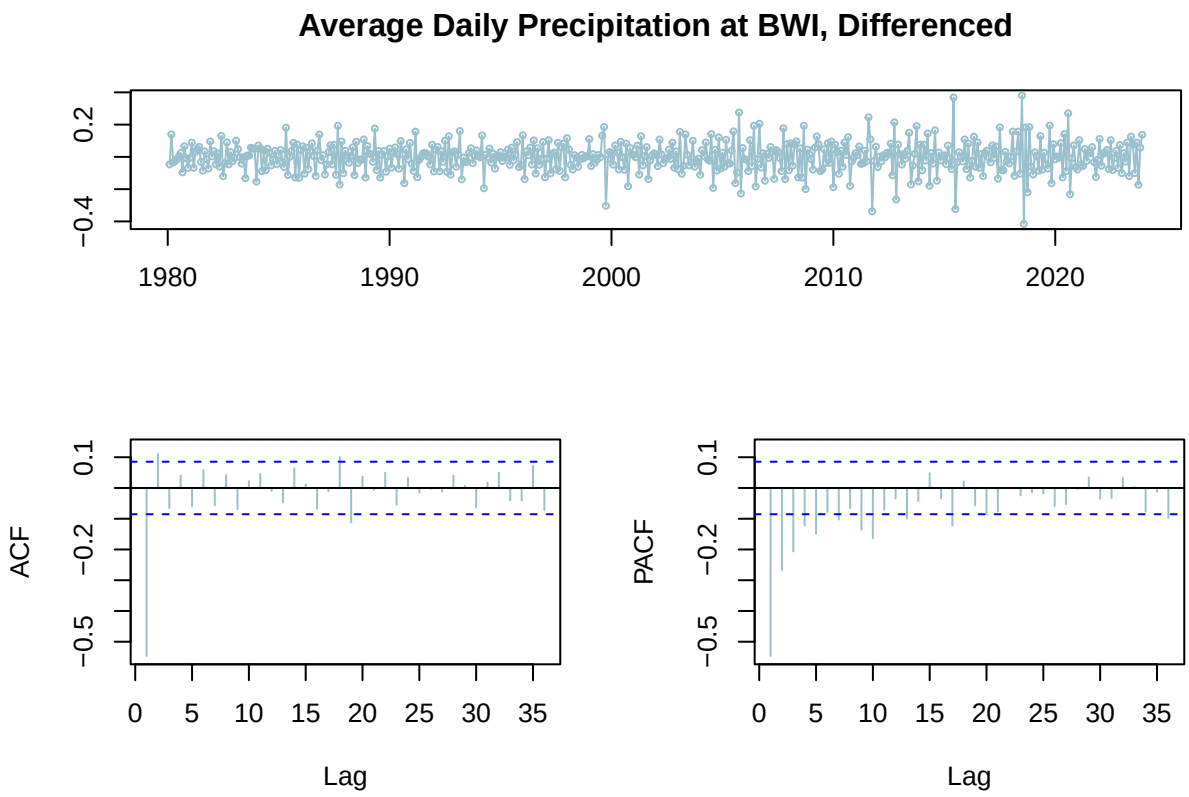
```
tsdisplay(precip, main = "Average Daily Precipitation at BWI", col = "lightblue3")
```



We see that there is a lot of variation in the average amount of monthly rainfall, with not much indication of many dynamics, although there seem to be significant spikes on the lags that are two months out on both the ACF and the PACF.

We check the differenced data as well, just be sure that the slight trend component visible doesn't change our results:

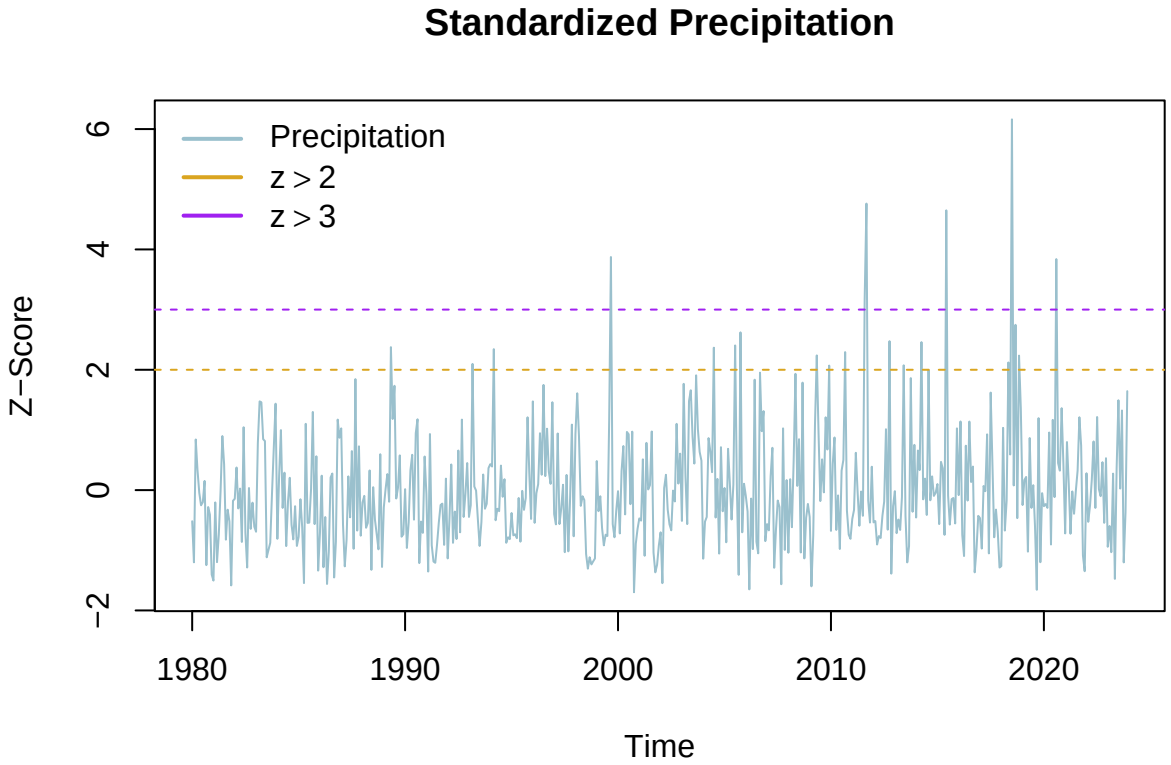
```
tsdisplay(diff(precip), main = "Average Daily Precipitation at BWI, Differenced",
          col = "lightblue3")
```



We see now that there is a significant spike at the first lag on the ACF and several significant spikes on the PACF, beyond those that would be expected from a pure MA-process caused decay. We see that spikes up through lag 10 appear to be significant on the PACF, with a couple higher order ones crossing the Bartlett windows in both the ACF and the PACF.

We standardize the precipitation plot to see if there are any outliers within the data:

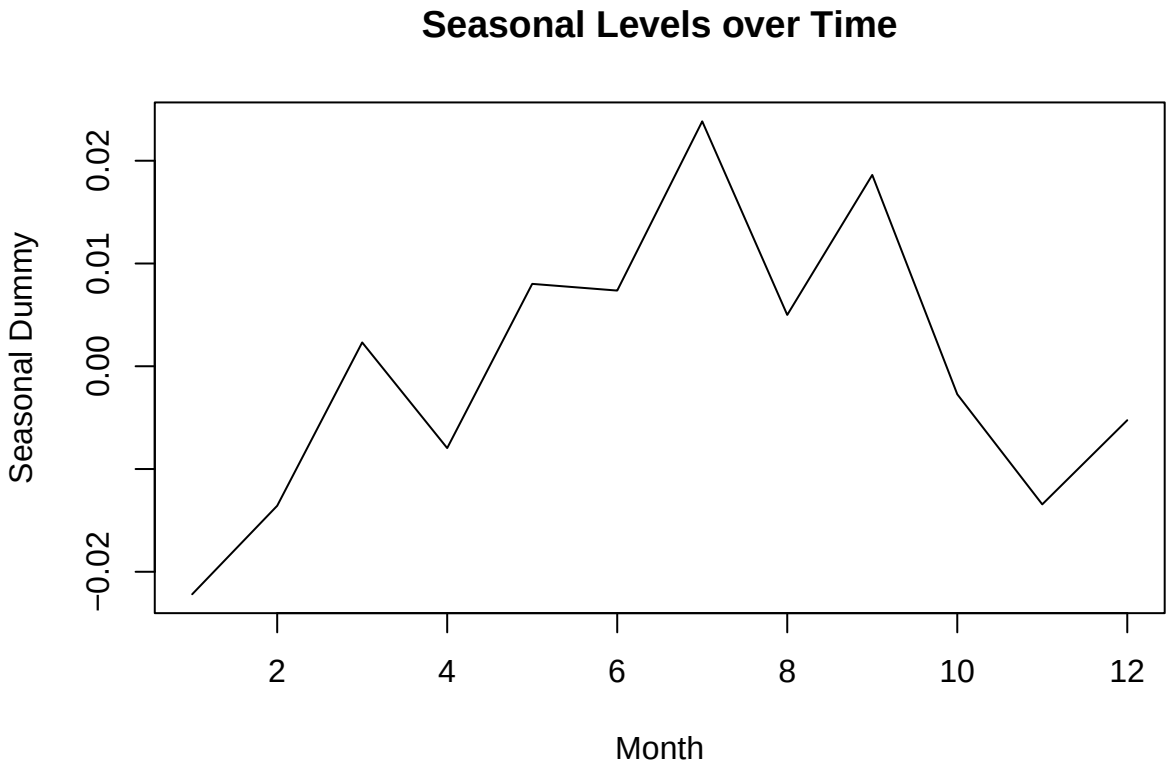
```
standardize <- function(x) {
  (x - mean(x)) / sd(x)
}

plot(standardize(precip), ylim = range(standardize(precip)), ylab = "Z-Score",
     main = "Standardized Precipitation", lty = 0)
legend("topleft", legend = c("Precipitation", expression(z > 2), expression(z > 3)),
       col = c("lightblue3", "goldenrod", "purple"), lwd = 2, lty = rep(1, 2, each = 2),
       box.lty = 0, text.width = 1)
lines(standardize(precip), col = "lightblue3")
abline(h = 2, col = "goldenrod", lty = 2)
abline(h = 3, col = "purple", lty = 2)
```



Note that the large spikes (which we define as $z > 2$) in the precipitation data occur more frequently as the time passes and that the very large spikes (which we define as $z > 3$) have been getting larger in magnitude over time as well. This is slightly problematic, as it means that a train test split will have the higher outliers in the test data, likely hurting the model.

Finally, we check for seasonality in the data:

```
plot(seasonal(stl(precip, s.window = "periodic"))[1:12], type = "l",
     ylab = "Seasonal Dummy", xlab = "Month", main = "Seasonal Levels over Time")
```



We see that the spikes in precipitation occur during the summer months (July through September), with the lowest precipitation occuring in January.

Precipitation Data Modelling

Setting Up the Train-Test Split

We first split the data into a training and testing component. It is unfortunate that the data got noisier as time went on, as this means that the later data (which will go into our test component) is likely to struggle to be well fit regardless. However, there's nothing that we can do about this, so we begin the process of creating the data.

We will do a 89-11 split. Since the precipitation data covers 44 years of time, this split gives 4 years worth of data to test over. Precipitation data is weather data, and forecasting more than a couple years out will be very unhelpful — we tried a 75-25 split and after 2 years or so, all the models stuck to their long-term case, as the data is just too complex to model after the first couple of years, so increasing the training size and having a shorter test window gives a sense of which models perform the best for the first couple years of forecasting, which is what we are going for anyways.

```
boundary <- as.integer(length(precip) * 10 / 11)
precip_train <- ts(precip[1:boundary], start = 1980, freq = 12)
precip_test <- ts(precip[(boundary+1):length(precip)], start = 1980 + boundary / 12,
                  freq = 12)
```

ARIMA Model

We first try the simplest model for the precipitation data — an ARIMA. The dynamics look rather difficult to determine, so we run `auto.arima` on the data, having manually told it to check up to lags of 10 for the AR component, which seemed to be the last significant ones in the differenced PACF. We left the maximum `q` as is, as the ACF indicates that past the first couple lags, none appear significant.

```
arima_train_mod <- auto.arima(precip_train, max.p = 10)
summary(arima_train_mod)
```

```
## Series: precip_train
## ARIMA(9,1,1)
##
## Coefficients:
##          ar1     ar2     ar3     ar4      ar5      ar6      ar7     ar8
##       0.0459  0.1323  0.0133  0.0108  -0.0369  -0.0064  -0.0533  -0.029
## s.e.  0.0459  0.0459  0.0462  0.0464   0.0463   0.0463   0.0463   0.046
##           ar9      ma1
##       -0.0600  -0.9878
## s.e.   0.0461   0.0063
##
## sigma^2 = 0.004616:  log likelihood = 611.57
## AIC=-1201.13   AICc=-1200.57   BIC=-1155.24
##
## Training set error measures:
##                       ME       RMSE        MAE       MPE     MAPE      MASE
## Training set 0.005107426 0.06715895 0.05030364 -54.74907 82.13042 0.7003372
##                      ACF1
## Training set -0.005833709
```

```
arima_train_fcst <- forecast(arima_train_mod, h = length(precip_test))
RMSE(arima_train_mod$residuals)
```

```
## [1] 0.06715895
```

```
RMSE(arima_train_fcst$mean - precip_test)
```

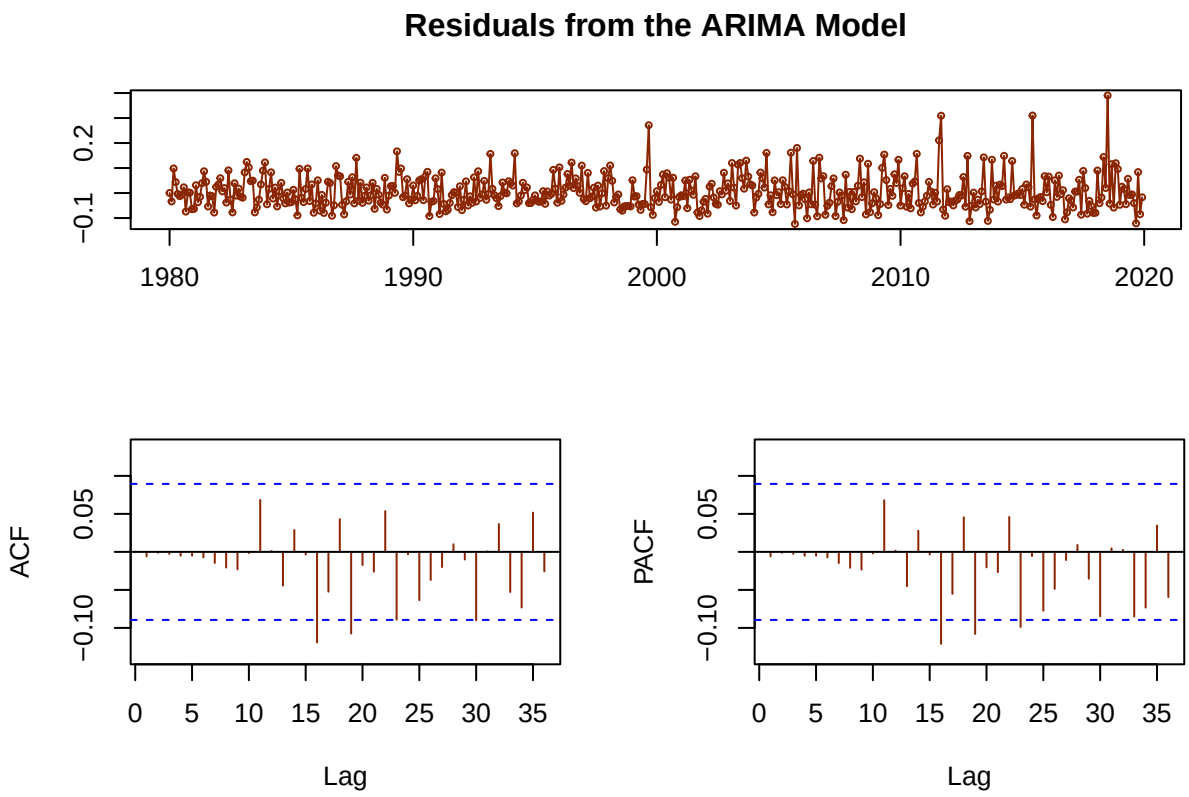
```
## [1] 0.06353472
```

We see that an ARIMA(9, 1, 1) model was proposed. This aligns with our initial observations from the differenced precipitation data, where similar numbers of lags were present in the ACF and PACF. Although there did not seem to be much of a trend in the data, it is good to know for reference that `auto.arima` preferred having the differencing of it; this implies that the trend was significant enough to affect the model.

We compute the RMSE for both the training and testing data and get values of 0.067 for the training data and 0.064 for the testing data. We note that the model gives an RMSE that is lower for the testing data than the training data. This tends to occur when the window for the test data is much shorter than that for the training data, which we have here, as there is a lot more data to build a model for (and imperfectly estimate) when looking at the training data than the testing data. However, we will only use these metrics to compare the models, so this is fine.

Now, we check the ARIMA residuals:

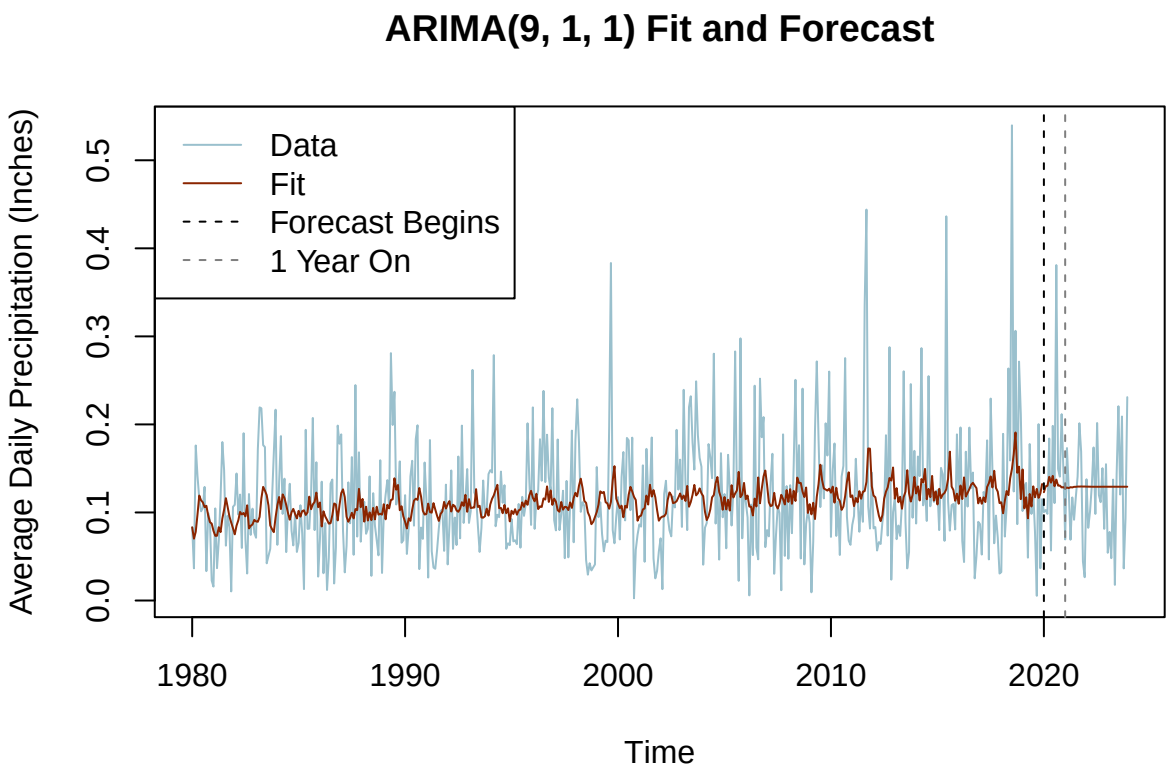
```
tsdisplay(arima_train_mod$residuals, main = "Residuals from the ARIMA Model",
          col = "orangered4")
```



We see that ARIMA has managed to account for most of the significant dynamics in the data, already, which is a good indicator that the model isn't too bad.

Finally, we plot the data, with the ARIMA model and its forecast fit to it:

```
plot(precip, col = "lightblue3", main = "ARIMA(9, 1, 1) Fit and Forecast",
     ylab = "Average Daily Precipitation (Inches)")
lines(fitted(arima_train_mod), col = "orangered4")
lines(arima_train_fcst$mean, col = "orangered4")
abline(v = 1980 + boundary / 12, col = "black", lty = 2)
abline(v = 1980 + boundary / 12 + 1, col = "gray50", lty = 2)
legend("topleft", legend = c("Data", "Fit", "Forecast Begins", "1 Year On"),
       lty = c(1, 1, 2, 2), col = c("lightblue3", "orangered4", "black", "gray50"))
```

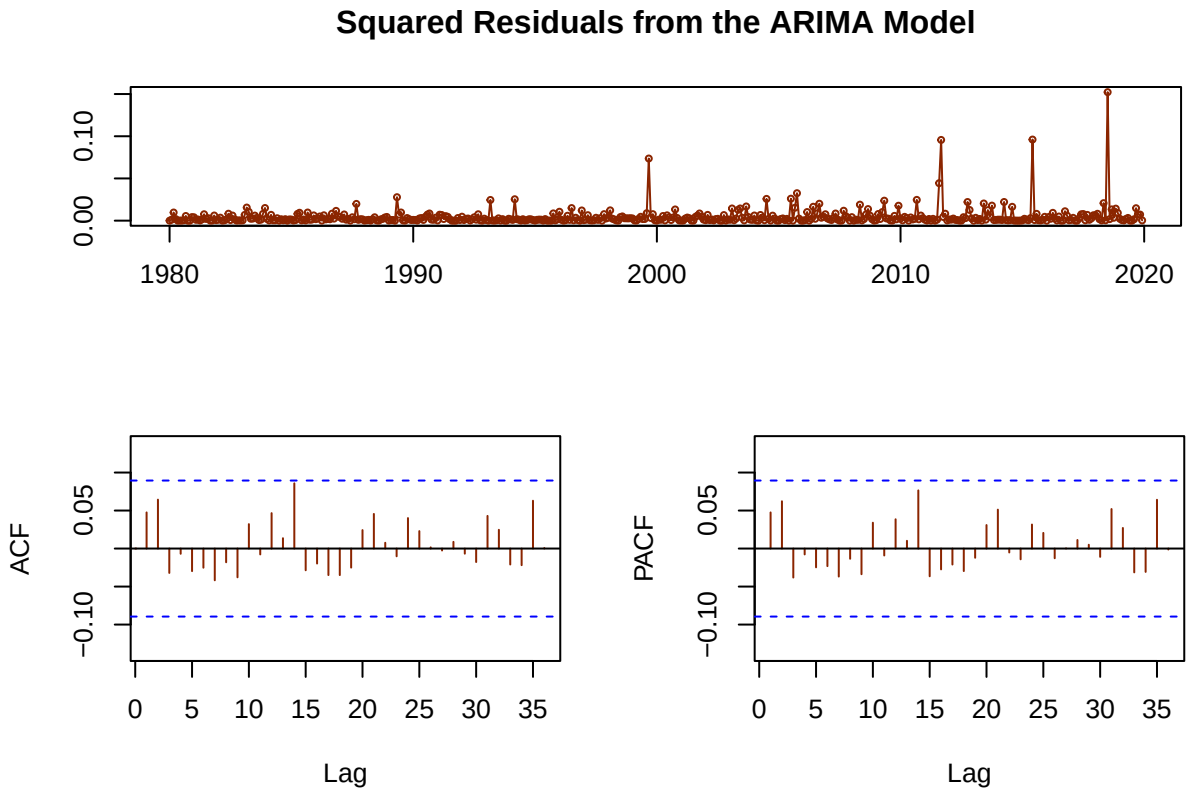


Although the error statistics did not look bad and it looked like the dynamics were accounted for, it is apparent that the ARIMA model is not great for modelling this data. It captures the general trend of the data, but misses the magnitude of fluctuations within the data. The forecast does as an ARIMA forecast is wont to do: it mean reverts after just over a year. This is fine, but we would rather have the forecast predict some further dynamics in the data, as this doesn't say much beyond that it expects the precipitation levels to go to the mean value.

ARCH Model

We now check the squared residuals from the ARIMA model:

```
tsdisplay(arima_train_mod$residuals^2, main = "Squared Residuals from the ARIMA Model",
          col = "orangered4")
```



We see that at the first lag displays some level of dynamics, and although this is not quite as pronounced as we may like, it indicates that an ARCH(1) model may help to capture a bit more of the dynamics. So we run it:

```
arch_framework <- ugarchspec(
  variance.model = list(model = "sGARCH", garchOrder = c(0, 1)),
  mean.model = list(armaOrder = c(9, 1), include.mean = TRUE),
  distribution.model = "sstd")
arch_train_mod <- ugarchfit(spec = arch_framework, data = precip_train)
arch_train_fcst <- ugarchforecast(arch_train_mod, data = NULL,
                                  n.ahead = length(precip_test), n.roll = 0,
                                  out.sample = 0)
t(arch_train_mod@model$pars[arch_train_mod@model$pars[, "Include"] == 1, "Level",
                            drop = FALSE])
```

```
##              mu       ar1        ar2        ar3      ar4          ar5
## Level 0.1170767 0.1297225 0.06801658 0.01126495 0.038027 -0.002444525
##              ar6         ar7         ar8          ar9         ma1        omega
## Level 0.01961799 -0.01562333 -0.00243087 -0.009789075 -0.07214569 0.0001415191
##           beta1     skew    shape
## Level 0.9673245 1.789206 7.478454
```

```
RMSE(arch_train_mod@fit$residuals)
```

```
## [1] 0.06765568
```

```
RMSE(arch_train_fcst@forecast$seriesFor - precip_test)
```

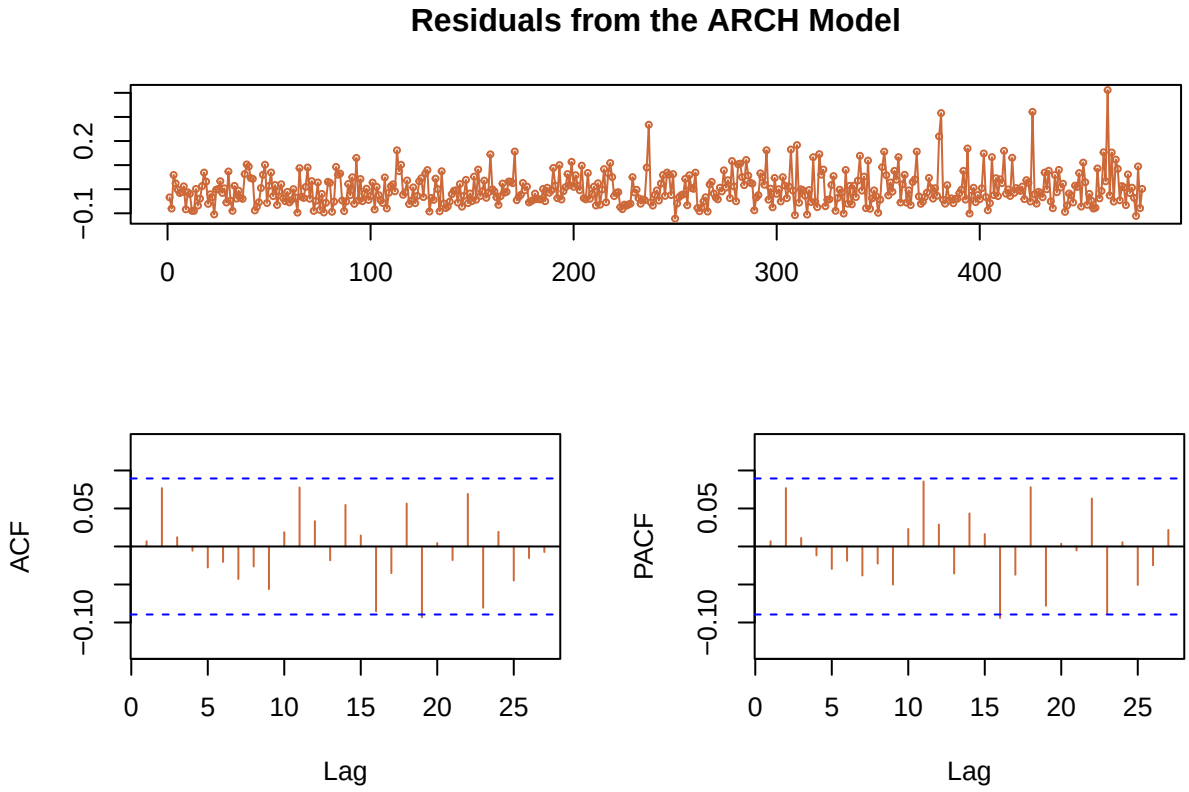
```
## [1] 0.0652269
```

We see that the ARMA parameters proposed are of similar levels to the ARIMA model, with the exception

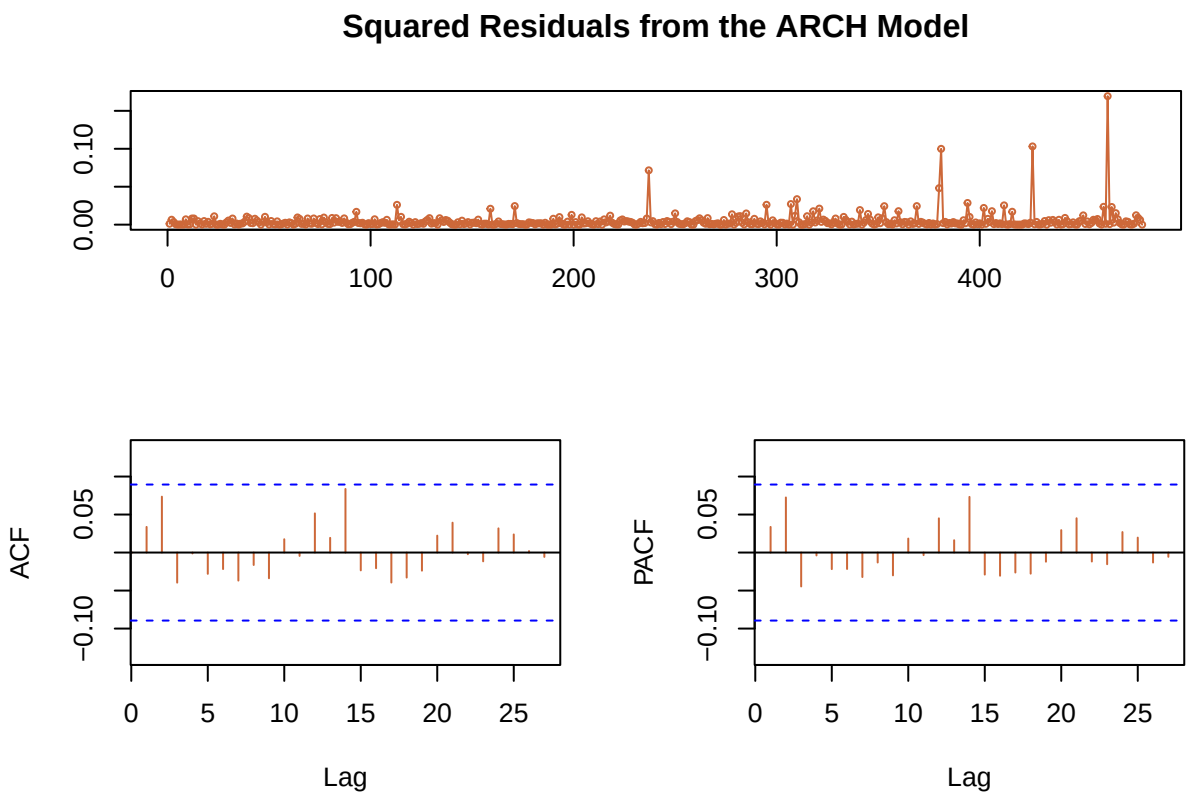
of the MA parameter, which is now two orders of magnitude more negative than the ARIMA MA parameter. Compared to the ARIMA model, ARCH seems to perform slightly worse in terms of the RMSE, with a training RMSE of 0.068 as compared to the ARIMA's 0.067, and a testing RMSE of 0.065 instead of ARIMA's 0.064 So it seems that the simpler ARIMA model is actually slightly better than the ARCH model for estimating the data.

However, seeing as we built the model, we will examine some plots to see how it performs.

```
tsdisplay(arch_train_mod@fit$residuals, main = "Residuals from the ARCH Model",
          col = "sienna3")
```

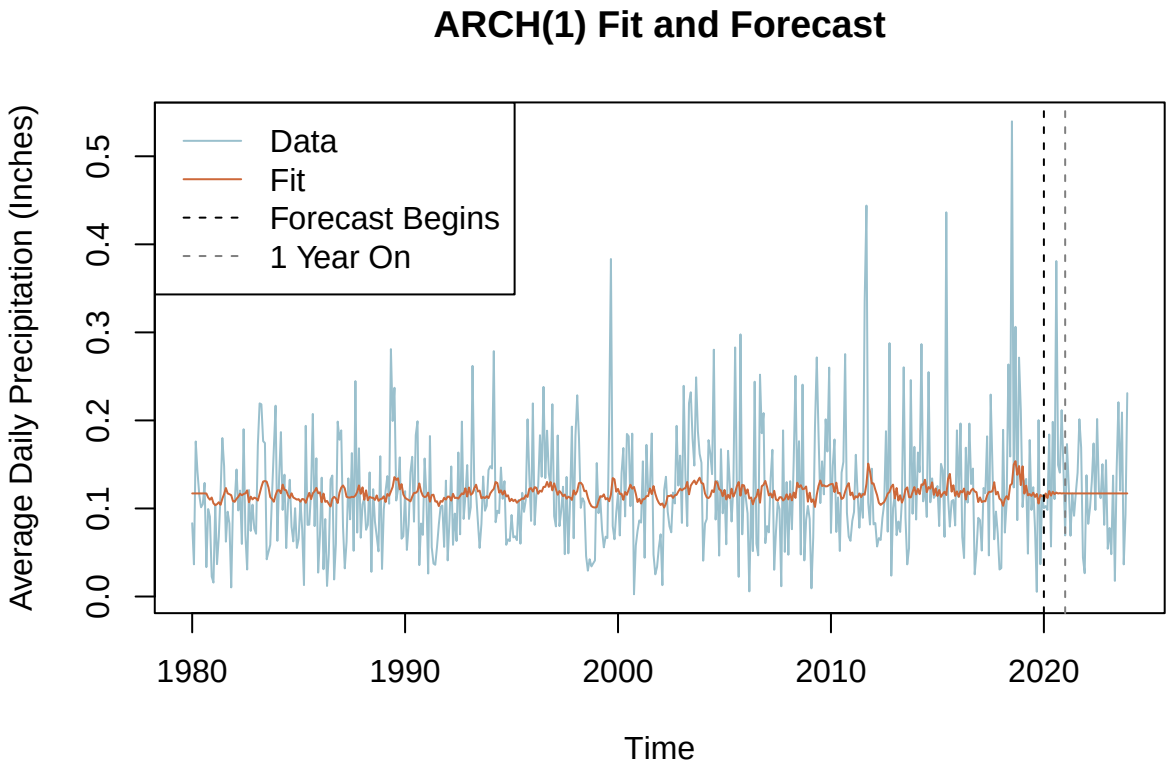



```
tsdisplay(arch_train_mod@fit$residuals^2, main = "Squared Residuals from the ARCH Model",
          col = "sienna3")
```



We see that no dynamics are present in the residuals or squared residuals of the ARCH model. So we can conclude that the ARCH(1) model improves the modelling of the volatility in the ARIMA. We now plot:

```
plot(precip, col = "lightblue3", main = "ARCH(1) Fit and Forecast",
     ylab = "Average Daily Precipitation (Inches)")
lines(as.ts(fitted(arch_train_mod), start = 1980), col = "sienna3")
lines(ts(arch_train_fcst@forecast$seriesFor, start = 1980 + boundary / 12,
         freq = 12), col = "sienna3")
abline(v = 1980 + boundary / 12, col = "black", lty = 2)
abline(v = 1980 + boundary / 12 + 1, col = "gray50", lty = 2)
legend("topleft", legend = c("Data", "Fit", "Forecast Begins", "1 Year On"),
       lty = c(1, 1, 2, 2), col = c("lightblue3", "sienna3", "black", "gray50"))
```



We see that the ARCH model has the same problem as the ARIMA — it consistently can match the direction the data is travelling but completely underestimates the magnitude of the data. The forecast also mean reverts much faster than the ARIMA does, indicating that it might be too complex for the data.

ETS Model

We now move on to building an ETS model for the data, as we suspect it may be able to capture the seasonal patterns of the data better.

```
ets_train_mod <- ets(precip_train)
summary(ets_train_mod)
```

```
## ETS(M,A,M)
##
## Call:
##  ets(y = precip_train)
##
##   Smoothing parameters:
##     alpha = 1e-04
##     beta  = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 0.1018
##     b = 0
##     s = 0.927 0.884 1.0266 1.2014 1.0167 1.1692
##            1.0847 1.0955 0.9121 1.0574 0.877 0.7484
##
##   sigma:  0.5525
##
##      AIC     AICc      BIC
```

```
## 344.6951 346.0198 415.6495
##
## Training set error measures:
##                          ME       RMSE        MAE       MPE     MAPE      MASE
## Training set -8.230451e-05 0.06670215 0.05092532 -64.10708 89.08194 0.7089922
##                     ACF1
## Training set 0.04929358
```

```
ets_train_fcst <- forecast(ets_train_mod, h = length(precip_test))

accuracy(ets_train_mod)["Training set", "RMSE"]
```

```
## [1] 0.06670215
```

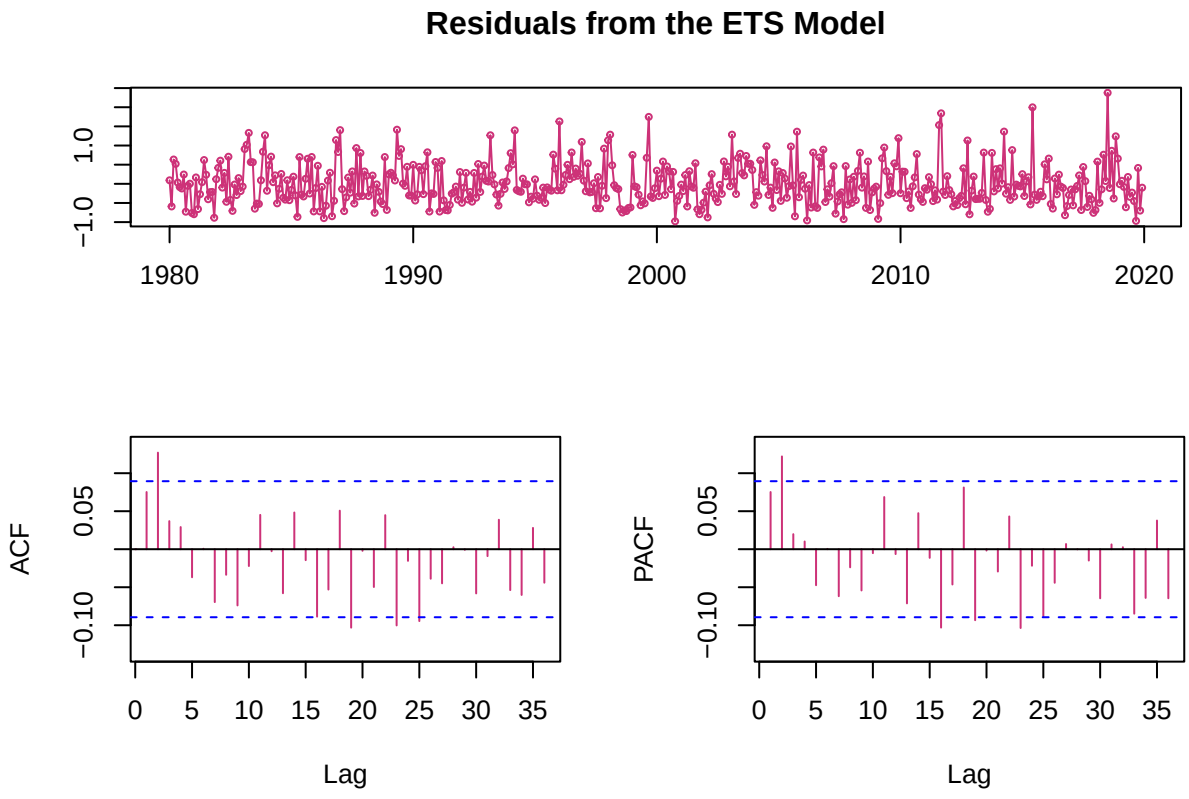
```
RMSE(ets_train_fcst$mean - precip_test)
```

```
## [1] 0.06393931
```

We see that the ETS model is proposed with multiplicative error, no trend, and multiplicative seasonality. The lack of trend is interesting, as this clashes with the suggestion from the ARIMA model, which desired a trend. However, the multiplicative seasonality proposed by the ETS model is in line with what we expect from looking at the standardized plot: we saw that the spikes were getting larger as time went by, so accounting for this in the seasonality component makes sense. Checking the RMSEs of the training and testing data, we see that they are essentially identical to what they were for the ARIMA model — for the training data: 0.067, and the testing: 0.064.

We now check the residuals of the model to see if ETS has captured all of the dynamics to the data.

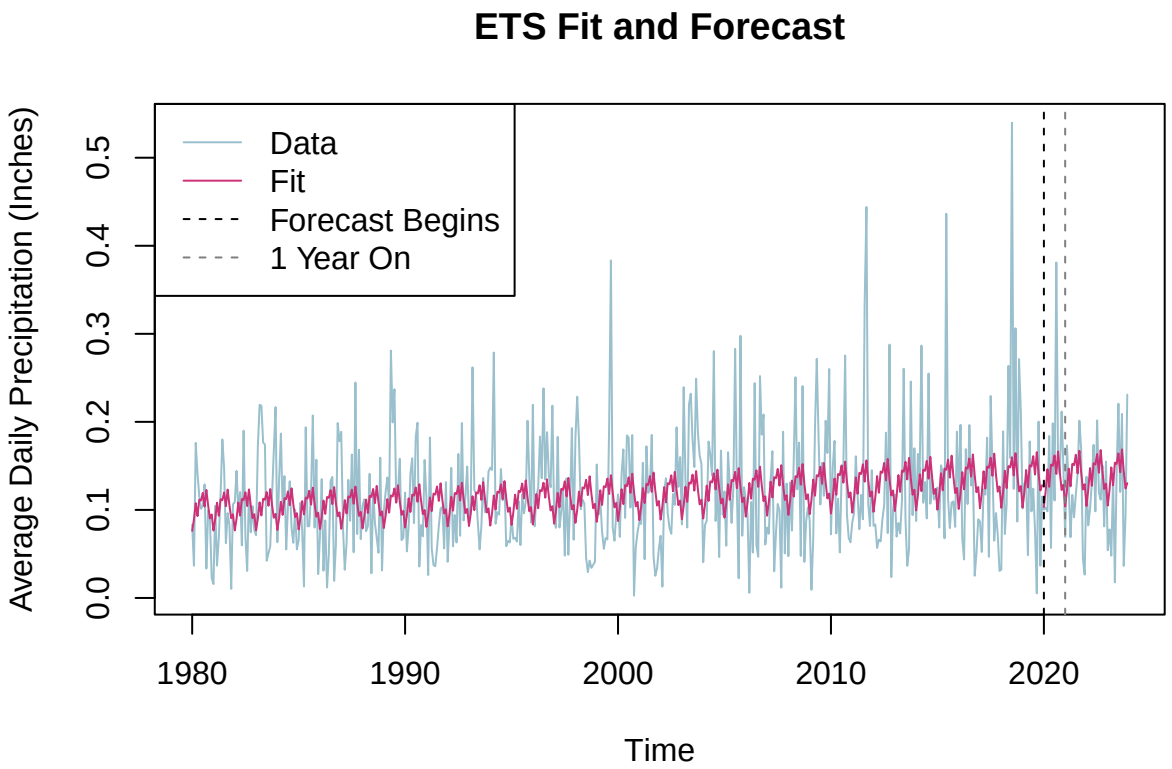
```
tsdisplay(ets_train_mod$residuals, main = "Residuals from the ETS Model",
          col = "violetred3")
```



We see that ETS seems to have done slightly worse at this task than ARIMA, with some high-level lags

present slightly clearing the Bartlett windows, but they are hardly significant. We can thus conclude that we have modelled most of the dynamics of the data with the ETS. Finally, we plot the model fits_train for ETS:

```
plot(precip, col = "lightblue3", main = "ETS Fit and Forecast",
     ylab = "Average Daily Precipitation (Inches)")
lines(fitted(ets_train_mod), col =  "violetred3")
lines(ets_train_fcst$mean, col =  "violetred3")
abline(v = 1980 + boundary / 12, col = "black", lty = 2)
abline(v = 1980 + boundary / 12 + 1, col = "gray50", lty = 2)
legend("topleft", legend = c("Data", "Fit", "Forecast Begins", "1 Year On"),
       lty = c(1, 1, 2, 2), col = c("lightblue3", "violetred3", "black", "gray50"))
```



ETS performs poorly in a different way to the ARIMA model. Even though it is using a multiplicative seasonality model, it still cannot account for all of the variation in the data, and when it is forecast out, we wind up matching the data cycles but not the magnitudes of the spikes. This is better than the ARIMA family models, as it continues to capture some dynamics to the data for the entire time, and accounts for seasonality over time, meaning that for long-run forecasting this is a decent model, as it captures the seasonal dynamics of the data in a way that the ARIMA ones don't. However, it seems to be overestimating the trend in the data, and if used for forecasting for too long appears to be likely to diverge from the values. However, for about a year out, the model does seem to capture limited dynamics of the data.

Holt-Winters

```
hw_train_mod <- hw(precip_train, h = length(precip_test))
summary(hw_train_mod$model)
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = precip_train, h = length(precip_test))
##
```

```
##   Smoothing parameters:
##     alpha = 0.0052
##     beta  = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 0.1113
##     b = 1e-04
##     s = -0.0051 -0.0191 -0.002 0.0246 0.0054 0.0291
##            0.0065 0.0088 -0.0109 0.0026 -0.0164 -0.0234
##
##   sigma:  0.0683
##
##      AIC     AICc      BIC
## 404.8269 406.1515 475.7812
##
## Training set error measures:
##                       ME       RMSE        MAE       MPE     MAPE      MASE
## Training set 0.0006483543 0.06716419 0.05094667 -62.52284 87.69277 0.7092895
##                    ACF1
## Training set 0.05660845
```

```
accuracy(hw_train_mod)["Training set", "RMSE"]
```

```
## [1] 0.06716419
```

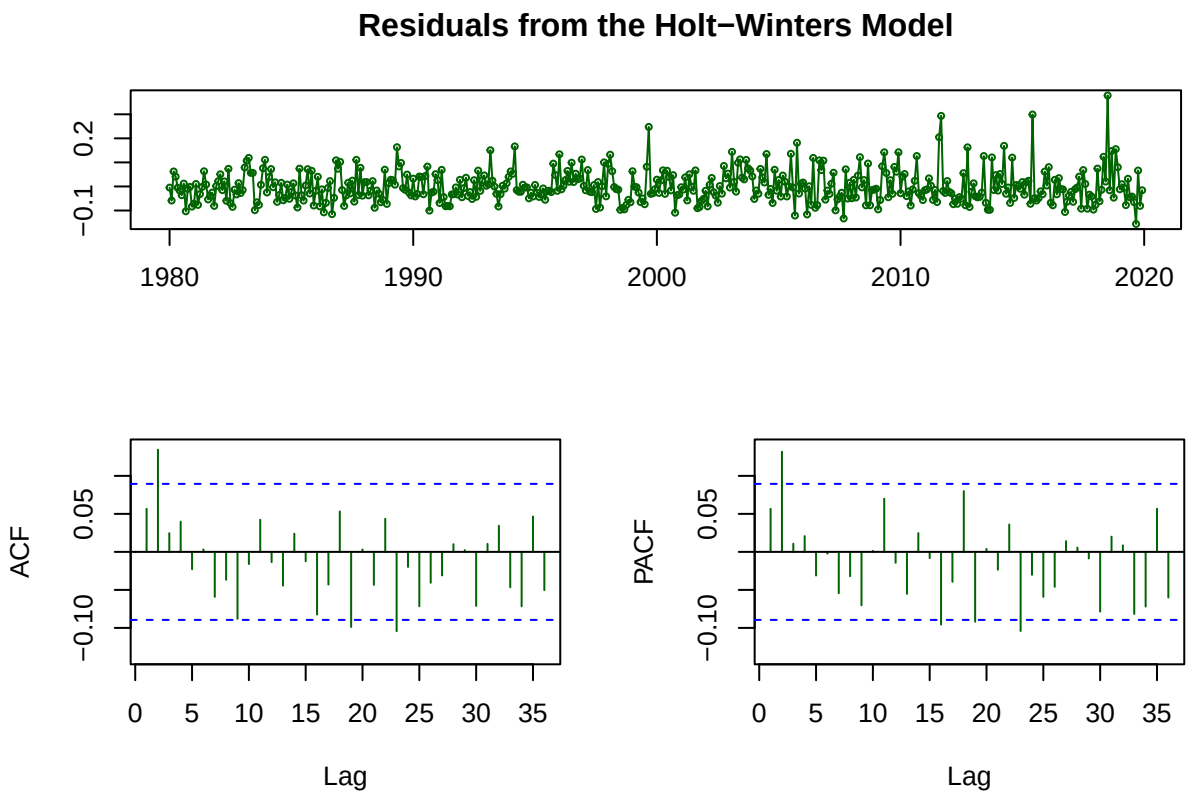
```
RMSE(hw_train_mod$mean - precip_test)
```

```
## [1] 0.06299494
```

We get similar RMSE values for the Holt-Winters model as we do for the ARIMA, ARCH, and ETS ones, although these are the best ones we've seen so far for the test data.

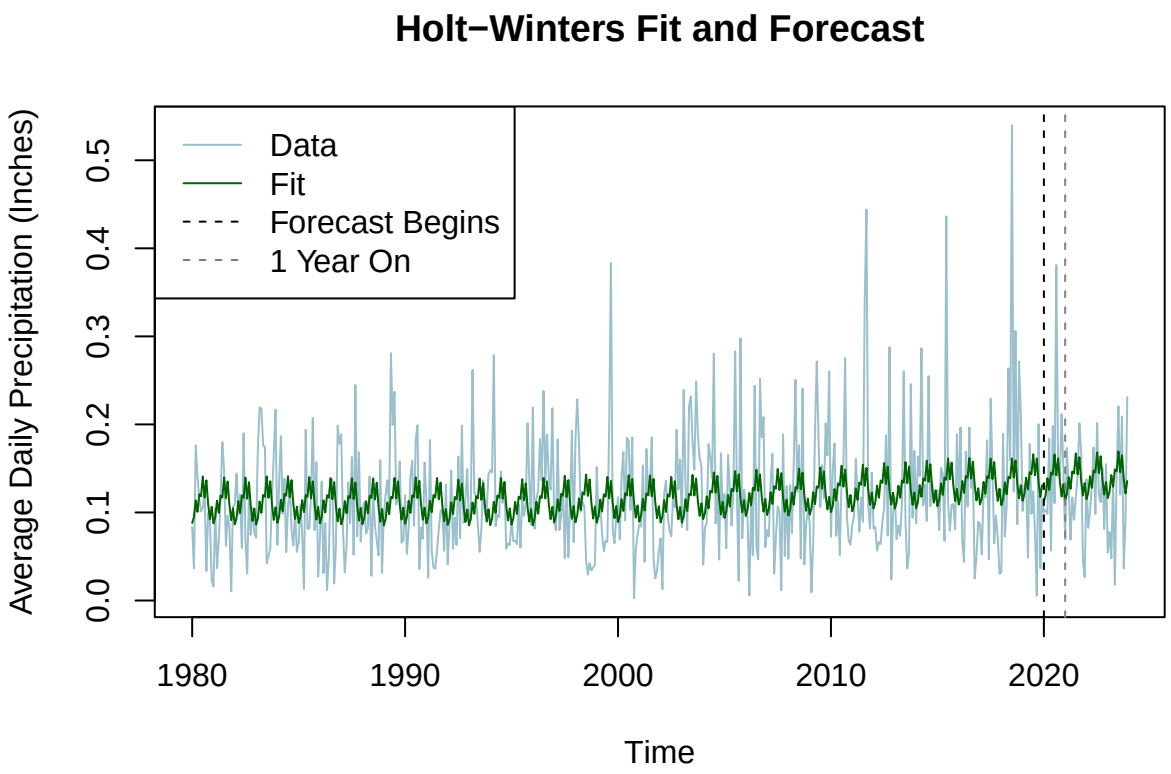
We check the residuals next:

```
tsdisplay(hw_train_mod$residuals, main = "Residuals from the Holt-Winters Model",
          col = "darkgreen")
```



The Holt-Winters model has, not shockingly, also managed to wipe away most of the dynamics that were present in the data. We see that the few lags that remain significant hardly clear the Bartlett window. Finally, we plot the data with the Holt-Winters fit and forecast:

```
plot(precip, col = "lightblue3", main = "Holt-Winters Fit and Forecast",
     ylab = "Average Daily Precipitation (Inches)")
lines(fitted(hw_train_mod), col = "darkgreen")
lines(hw_train_mod$mean, col = "darkgreen")
abline(v = 1980 + boundary / 12, col = "black", lty = 2)
abline(v = 1980 + boundary / 12 + 1, col = "gray50", lty = 2)
legend("topleft", legend = c("Data", "Fit", "Forecast Begins", "1 Year On"),
       lty = c(1, 1, 2, 2), col = c("lightblue3", "darkgreen", "black", "gray50"))
```



We see that the Holt-Winters model provides a very similar fit to the data as the ETS does, which makes sense, as Holt-Winters is an ETS model with the parameters already specified. It doesn't do as well, as it seems to be on track to overestimate the final data.

NNETAR

So far, the models from the ARIMA and ETS families all seem to have the same problem: while they are able to capture the trend of the data decently, the magnitudes remain rather muted and their forecasts aren't very interesting. We want to see if some more advanced methods can do a better job in forecasting, although there is a concern that because this data has so few dynamics, they might overfit the trainng data and completely miss the testing data. But first, we try an NNETAR model to see what happens.

```
nnet_train_mod <- model(as_tsibble(precip_train), NNETAR(value))
nnet_train_fcst <- forecast(nnet_train_mod, simulate = FALSE, h = length(precip_test))
RMSE(residuals(nnet_train_mod)$.resid)
```

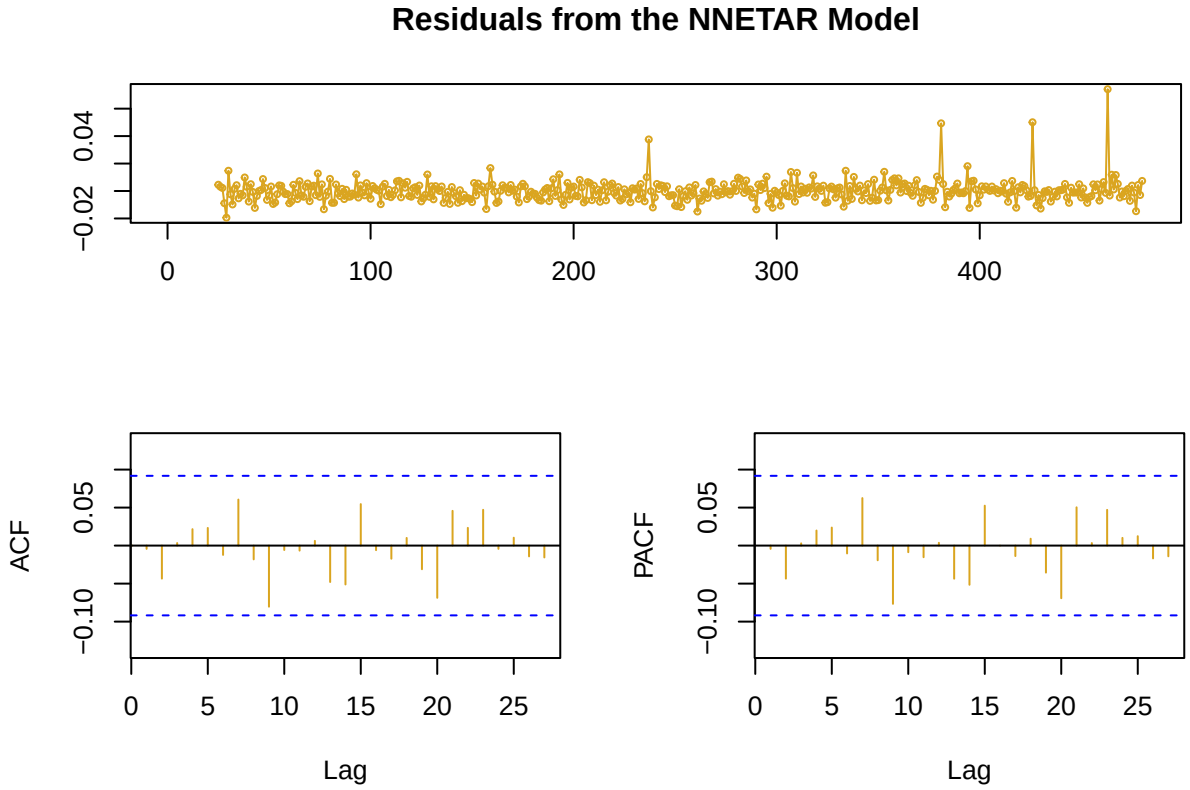
```
## [1] 0.007638513
```

```
RMSE(nnet_train_fcst$.mean - precip_test)
```

```
## [1] 0.07560244
```

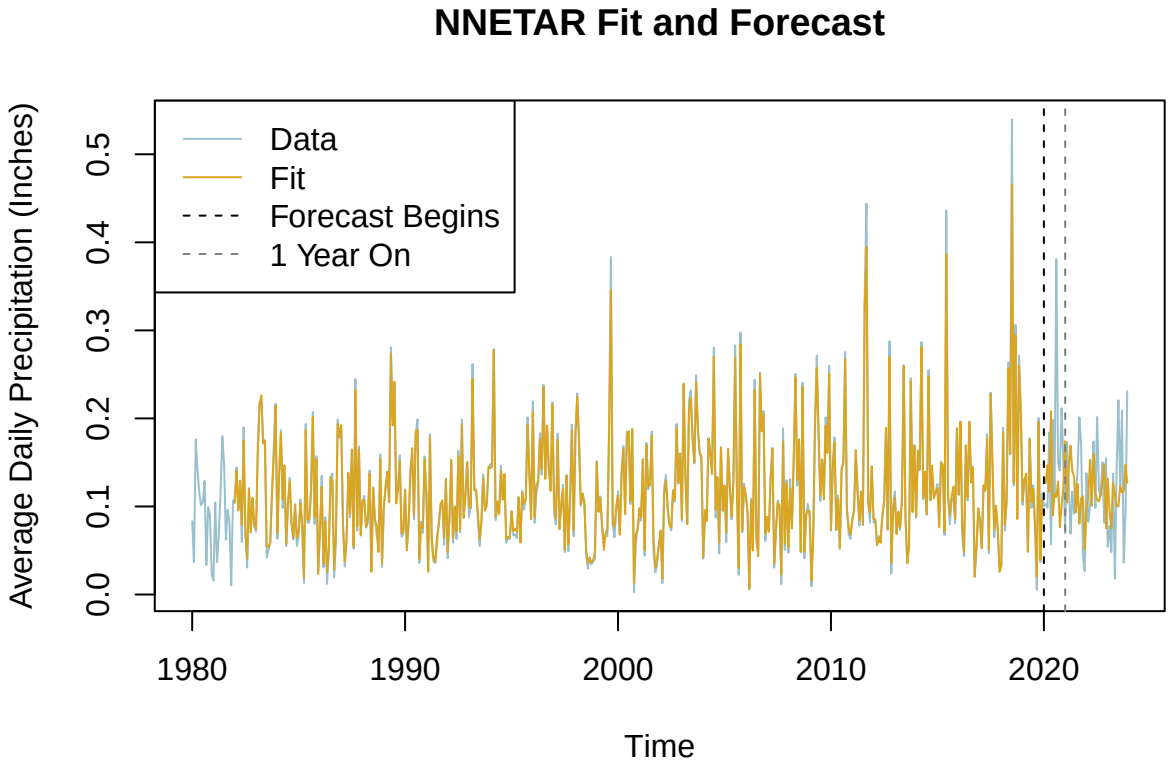
This model fits_train the training data extremely well: we have an RMSE for the training data of 0.008, but unfortunately struggles with the test data, giving the highest RMSE so far of: 0.076. This suggests that the NNETAR model is overfitting the data.

```
tsdisplay(residuals(nnet_train_mod)$.resid, col = "goldenrod",
          main = "Residuals from the NNETAR Model")
```



We see that the NNETAR model has managed to remove all significant dynamics from the residuals — indicating that it fits_train the training data extremely well, which the RMSE suggested as well.

```
plot(precip, col = "lightblue3", main = "NNETAR Fit and Forecast",
     ylab = "Average Daily Precipitation (Inches)")
lines(ts(fitted(nnet_train_mod)$.fitted, start = 1980, freq = 12), col = "goldenrod")
lines(ts(nnet_train_fcst$.mean, start = 1980 + boundary / 12, freq = 12),
      col = "goldenrod")
abline(v = 1980 + boundary / 12, col = "black", lty = 2)
abline(v = 1980 + boundary / 12 + 1, col = "gray50", lty = 2)
legend("topleft", legend = c("Data", "Fit", "Forecast Begins", "1 Year On"),
       lty = c(1, 1, 2, 2), col = c("lightblue3", "goldenrod", "black", "gray50"))
```



At last, we have managed to fit to the data a model that can capture most of its dynamics. We see that it matches the magnitudes better than other models that we have tried. However, the model severely underestimates the massive spike that occurs in the data for the year after the testing data occurred. After that, it seems to capture the rest of the dynamics decently well, indicating that perhaps the high test RMSE is caused by the data behaving oddly in this spot.

Prophet

Finally, we try the Prophet model, as our data is quite nightmarish, so there may be some unknown complex seasonal processes underlying it. We try multiplicative seasonality for this data, as that is the trend detected by the ETS model:

```
prophet_train_mod <- model(as_tsibble(precip_train),
                           prophet(value ~ season(period = 12, type = "multiplicative")))
prophet_train_fcst <- forecast(prophet_train_mod, h = length(precip_test))
RMSE(residuals(prophet_train_mod)$.resid)
```

```
## [1] 0.06650653
```

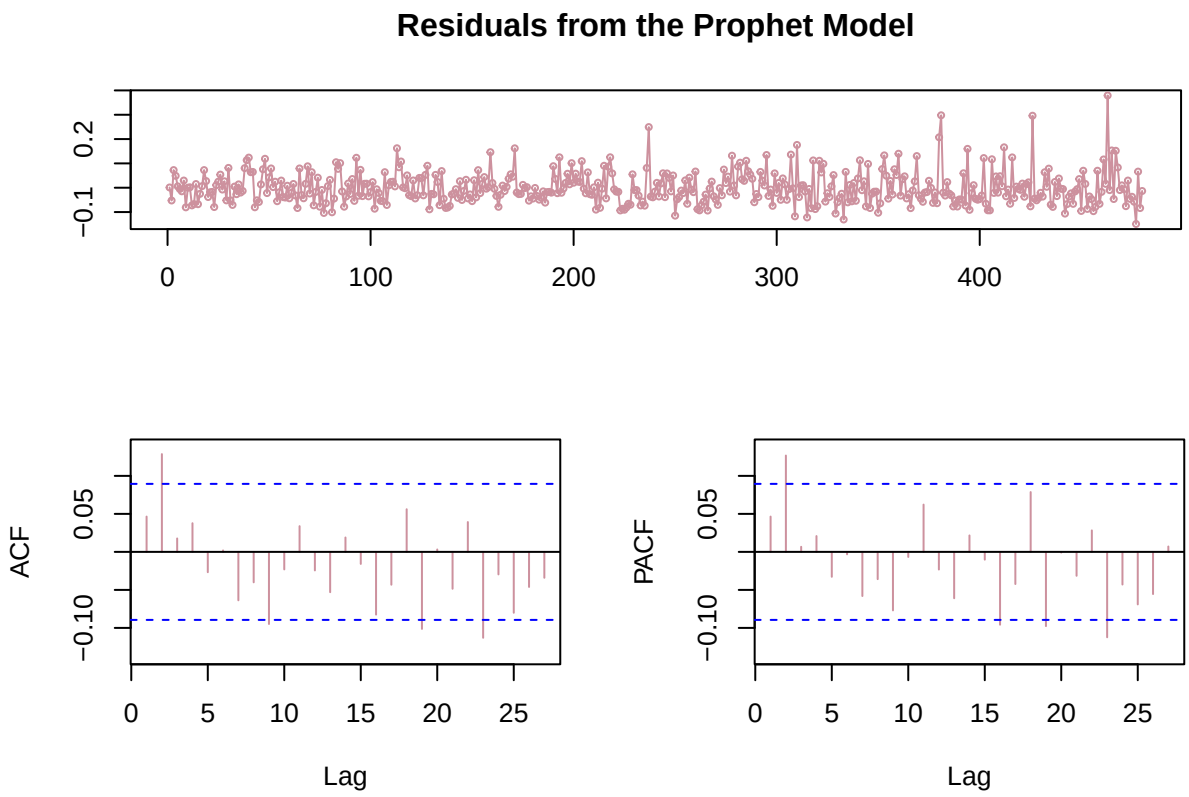
```
RMSE(prophet_train_fcst$.mean - precip_test)
```

```
## [1] 0.06366091
```

This model fits_train the training data in a similar manner to the ARIMA and ETS models: we have an RMSE for the training data of 0.067 and giving an RMSE for the test data of: 0.064.

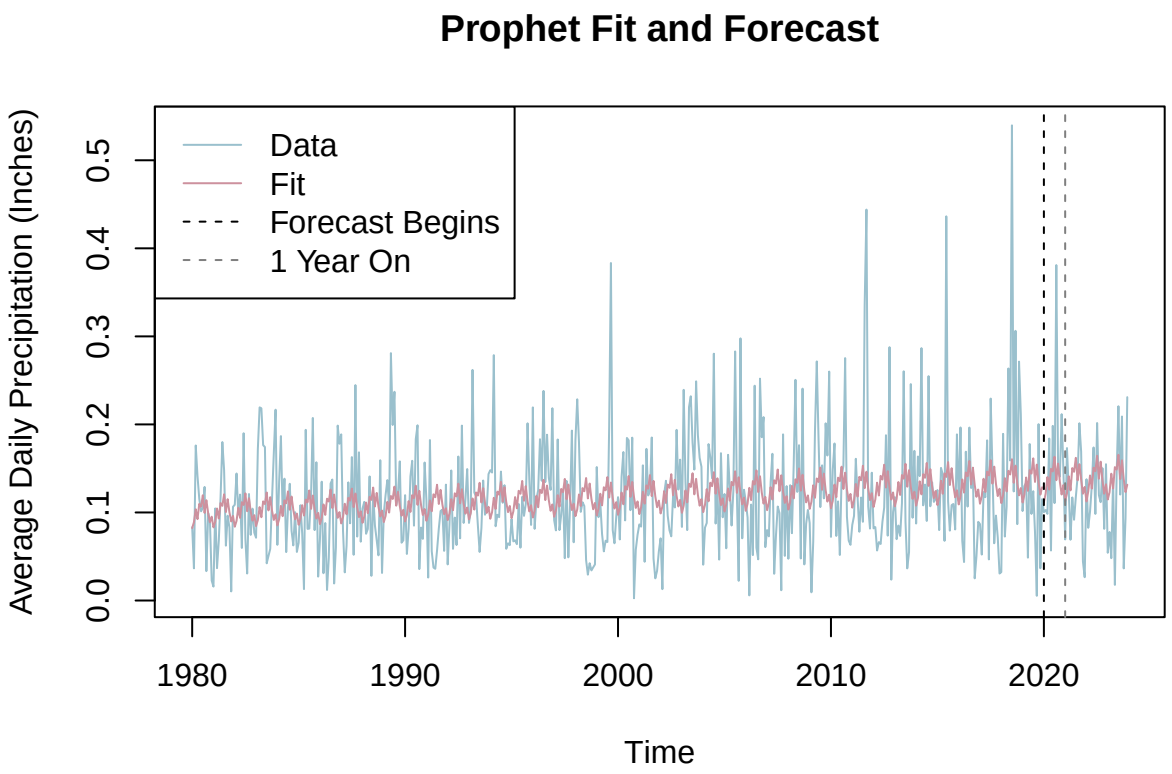
We check the residuals plot:

```
tsdisplay(residuals(prophet_train_mod)$.resid, col = "pink3",
          main = "Residuals from the Prophet Model")
```



We see that the Prophet model has managed to almost completely erase any dynamics from the residuals. This is good, it indicates that the model is struggling because the data has few dynamics not because the model is not capturing them. We are confident now to go on and analyze the plots of the data and its forecast:

```
plot(precip, col = "lightblue3", main = "Prophet Fit and Forecast",
     ylab = "Average Daily Precipitation (Inches)")
lines(ts(fitted(prophet_train_mod)$.fitted, start = 1980, freq = 12), col = "pink3")
lines(ts(prophet_train_fcst$.mean, start = 1980 + boundary / 12, freq = 12),
      col = "pink3")
abline(v = 1980 + boundary / 12, col = "black", lty = 2)
abline(v = 1980 + boundary / 12 + 1, col = "gray50", lty = 2)
legend("topleft", legend = c("Data", "Fit", "Forecast Begins", "1 Year On"),
       lty = c(1, 1, 2, 2), col = c("lightblue3", "pink3", "black", "gray50"))
```



We see that Prophet manages to capture an upwards trend in the data and has some seasonality in it, like the ETS. It appears to match the seasonal behavior of the future data values as well.

Combination

We now test combinations of the model on our data, hunting the best mix of parameters to maximize the RMSE. We want to optimize the weighted average of our data to have the smallest RMSE on the residuals over a rolling window:

```
fits_train <- cbind(fitted(arima_train_mod), as.ts(fitted(arch_train_mod), start = 1980),
                    fitted(ets_train_mod), fitted(hw_train_mod),
                    ts(c(as.numeric(apply(cbind(fitted(arima_train_mod),
                                                as.ts(fitted(arch_train_mod),
                                                      start = 1980),
                                                fitted(ets_train_mod),
                                                fitted(hw_train_mod),
                                                ts(fitted(prophet_train_mod)$.fitted,
                                                   start = 1980, freq = 12))[1:24,],
                                          1, mean)),
                         fitted(nnet_train_mod)$.fitted[-(1:24)]), start = 1980, freq = 12),
                    ts(fitted(prophet_train_mod)$.fitted, start = 1980, freq = 12))

forecasts_train <- cbind(arima_train_fcst$mean, arch_train_fcst@forecast$seriesFor,
                         ets_train_fcst$mean, hw_train_mod$mean, nnet_train_fcst$.mean,
                         prophet_train_fcst$.mean)
weights_train <- rep(1/6, 6)

for (k in seq_len(length(precip_train))) {
  for (i in seq_len(6)) {
    weights_pos <- weights_neg <- weights_train
```

```
    weights_pos[i] <- weights_train[i] + 0.001
    weights_neg[i] <- weights_train[i] - 0.001
    weights_new <- weights_train
    if (RMSE(precip[1:k] - fits_train[1:k,] %*% weights_train) >
        RMSE(precip[1:k] - fits_train[1:k,] %*% weights_pos)) {
      weights_new <- weights_pos
    } else if (RMSE(precip[1:k] - fits_train[1:k,] %*% weights_train) >
               RMSE(precip[1:k] - fits_train[1:k,] %*% weights_neg)) {
      weights_new <- weights_neg
    }
    weights_train <- weights_new
  }
}
weights_train
```

```
## [1]  0.14966667 -0.03233333  0.23666667 -0.11133333  0.60666667  0.15466667
```

```
RMSE(precip_train - fits_train %*% weights_train)
```

```
## [1] 0.03071883
```

```
RMSE(precip_test - forecasts_train %*% weights_train)
```

```
## [1] 0.06880479
```

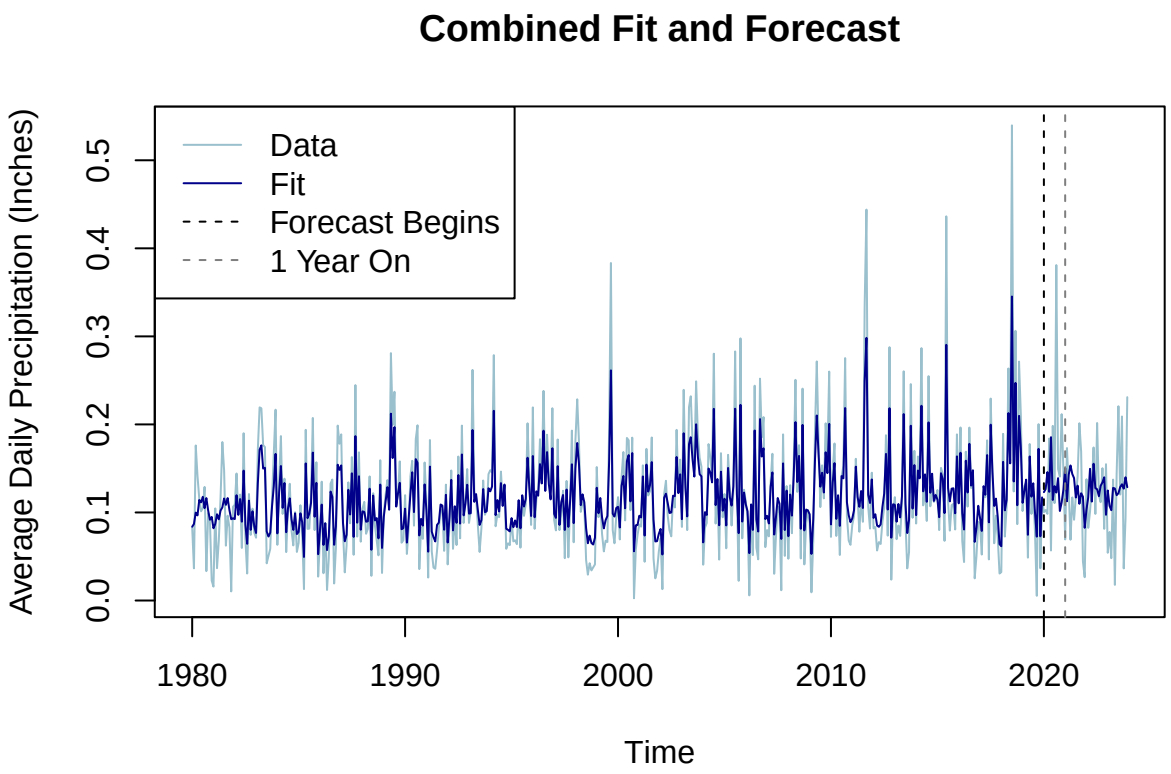
We find that the optimal forecast is the linear combination of the forecasts:

$$0.15 \times \text{ARIMA} - 0.032 \times \text{ARCH} - 0.237 \times \text{ETS} + 0.111 \times \text{HW} - 0.607 \times \text{NNETAR} - 0.155 \times \text{Prophet}$$

It has a much smaller train RMSE than any of the other data values, but still has a similar testing RMSE to the best forecasts of: 0.0688048, which makes sense, seeing as a combination of several forecasts with a similar RMSE won't improve the RMSE unless they were under- and overestimating the data by similar magnitudes, which is unlikely. This indicates that the data was already very well modeled by the other forecasts.

We now plot the fitted values and the expected forecasts on the data:

```
plot(precip, col = "lightblue3", main = "Combined Fit and Forecast",
     ylab = "Average Daily Precipitation (Inches)")
lines(ts(fits_train %*% weights_train, start = 1980, freq = 12), col = "blue4")
lines(ts(forecasts_train %*% weights_train, start = 1980 + boundary / 12, freq = 12),
      col = "blue4")
abline(v = 1980 + boundary / 12, col = "black", lty = 2)
abline(v = 1980 + boundary / 12 + 1, col = "gray50", lty = 2)
legend("topleft", legend = c("Data", "Fit", "Forecast Begins", "1 Year On"),
       lty = c(1, 1, 2, 2), col = c("lightblue3", "blue4", "black", "gray50"))
```



We see that this seems to do the best job of fitting the model out of all the models, managing to account for the trends in the forecasted data the best.

Analyzing the Forecasts

We quickly check the testing RMSEs of all of our forecasts to see if any should be removed:

```
cat("Forecast    Train RMSE    Test RMSE",
    "\nARIMA      ", round(RMSE(arima_train_mod$residuals), 5),
    "       ", round(RMSE(arima_train_fcst$mean - precip_test), 5),
    "\nARCH       ", round(RMSE(arch_train_mod@fit$residuals), 5), "       ",
    round(RMSE(arch_train_fcst@forecast$seriesFor - precip_test), 5),
    "\nETS        ", paste0(round(accuracy(ets_train_mod)["Training set", "RMSE"], 5), 0),
    "       ", round(RMSE(ets_train_fcst$mean - precip_test), 5),
    "\nH-W        ", round(accuracy(hw_train_mod)["Training set", "RMSE"], 5),
    "       ", round(RMSE(hw_train_mod$mean - precip_test), 5),
    "\nNNETAR     ", paste0(round(RMSE(residuals(nnet_train_mod)$.resid), 5), 0),
    "       ", round(RMSE(nnet_train_fcst$.mean - precip_test), 5),
    "\nProphet    ", round(RMSE(residuals(prophet_train_mod)$.resid), 5),
    "       ", round(RMSE(prophet_train_fcst$.mean - precip_test), 5),
    "\nCombo      ", round(RMSE(precip_train - fits_train %*% weights_train), 5),
    "       ", round(RMSE(precip_test - forecasts_train %*% weights_train), 5))
```

```
## Forecast    Train RMSE    Test RMSE
## ARIMA       0.06716         0.06353
## ARCH        0.06766         0.06523
## ETS         0.06670         0.06394
## H-W         0.06716         0.06299
## NNETAR      0.007640         0.0756
## Prophet     0.06651         0.06366
## Combo       0.03072         0.0688
```

We see that all of the models performed rather similarly, with test RMSEs of around 0.063. The only outliers that we have are the NNETAR and ARCH models, which have slightly higher test RMSEs. There is a decent chance, however, that the NNETAR model, which has the second lowest training RMSE and is counted very highly in the combination, does perform very well on the data and was only messed up by the issue with the spike in the testing data right at the start that it was unable to account for. We therefore choose to continue working with it. On the other hand, when we examined the forecasts of the ARCH model, we saw that it added barely any predictive value to the data. Additionally, it has the highest train RMSE, so we choose to remove the ARCH model from our forecast combination, as it only hurts the forecast.

Forecasting Unknown Values

We now build all of the models one last time on the full data set and forecast two years into the future, using all of the models that we developed in the combination except for the ARCH model.

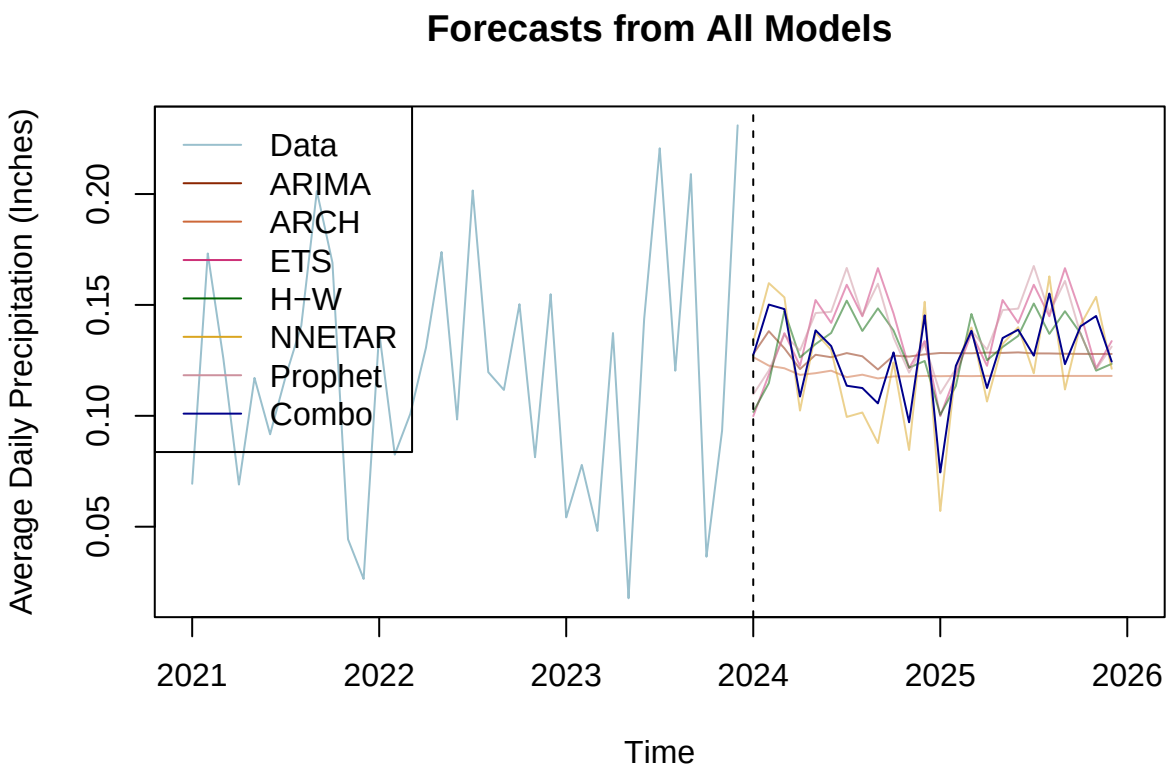
```
arima_mod <- auto.arima(precip, max.p = 10)
arima_fcst <- forecast(arima_mod, h = 24)
arch_mod <- ugarchfit(spec = arch_framework, data = precip)
arch_fcst <- ugarchforecast(arch_mod, data = NULL, n.ahead = 24, n.roll = 0,
                            out.sample = 0)
ets_mod <- ets(precip)
ets_fcst <- forecast(ets_mod, h = 24)
hw_mod <- hw(precip, h = 24)
nnet_mod <- model(as_tsibble(precip), NNETAR(value))
nnet_fcst <- forecast(nnet_mod, simulate = FALSE, h = 24)
prophet_mod <- model(as_tsibble(precip),
                     prophet(value ~ season(period = 12, type = "multiplicative")))
prophet_fcst <- forecast(prophet_mod, h = 24)

fits <- cbind(fitted(arima_mod), fitted(ets_mod), fitted(hw_mod),
              ts(c(as.numeric(apply(cbind(fitted(arima_mod), fitted(ets_mod),
                                          fitted(hw_mod), ts(fitted(prophet_mod)$.fitted,
                                                             start = 1980,
                                                             freq = 12))[1:24,],
                                    1, mean)), fitted(nnet_mod)$.fitted[-(1:24)]),
                 start = 1980, freq = 12),
              ts(fitted(prophet_mod)$.fitted, start = 1980, freq = 12))

forecasts <- cbind(arima_fcst$mean, ets_fcst$mean, hw_mod$mean, nnet_fcst$.mean,
                   prophet_fcst$.mean)

weights <- rep(1/5, 5)

for (k in seq_len(length(precip))) {
  for (i in seq_len(5)) {
    weights_pos <- weights_neg <- weights
    weights_pos[i] <- weights[i] + 0.001
    weights_neg[i] <- weights[i] - 0.001
    weights_new <- weights
    if (RMSE(precip[1:k] - fits[1:k,] %*% weights) >
        RMSE(precip[1:k] - fits[1:k,] %*% weights_pos)) {
      weights_new <- weights_pos
    } else if (RMSE(precip[1:k] - fits[1:k,] %*% weights) >
               RMSE(precip[1:k] - fits[1:k,] %*% weights_neg)) {
      weights_new <- weights_neg
```

```
    }
    weights <- weights_new
  }
}

weights
```

```
## [1]  0.144  0.161  0.025  0.692 -0.019
```

```
combo_forecast <- forecasts %*% weights
```

```
precip_small <- precip[(length(precip) - 35):length(precip)] %>%
  ts(frequency = 12, start = 2021)
plot(precip_small, col = "lightblue3", xlim = c(2021, 2026),
     main = "Forecasts from All Models",
     ylab = "Average Daily Precipitation (Inches)")
lines(arima_fcst$mean, col = alpha("orangered4", 0.5))
lines(ts(arch_fcst@forecast$seriesFor, start = 2024, freq = 12),
      col = alpha("sienna3", 0.5))
lines(ets_fcst$mean, col =  alpha("violetred3", 0.5))
lines(hw_mod$mean, col = alpha("darkgreen", 0.5))
lines(ts(nnet_fcst$.mean, start = 2024, freq = 12),
      col = alpha("goldenrod", 0.5))
lines(ts(prophet_fcst$.mean, start = 2024, freq = 12),
      col = alpha("pink3", 0.5))
lines(ts(forecasts %*% weights, start = 2024, freq = 12), col = "blue4")
abline(v = 2024, lty = 2)
legend("topleft", legend = c("Data", "ARIMA", "ARCH", "ETS", "H-W", "NNETAR",
                             "Prophet", "Combo"), lty = c(rep(1, 8), 2),
       col = c("lightblue3", "orangered4", "sienna3", "violetred3", "darkgreen",
               "goldenrod", "pink3", "blue4"))
```



We see that the final combination forecast tries to account for some level of seasonality in the data, while staying around the mean value of 0.11 or so, as predicted by the ARCH model.

III. Conclusions and Future Work

We can make a couple conclusions about the data. We see that we expect precipitation to vary around the mean value of around 0.11. Additionally, in the next two months, the precipitation looks likely to decrease followed by a small increase to a lower level than there was before. All the models seem to agree with this trend. However, after that, the models diverge from each other and it is difficult to know what will happen.

However, the main problem with all of the models is that they consistently underestimate the magnitude of the data. This is likely because the data is not very correlated with previous values, so forecasts and model fits tend to stay around the mean value with small deviations to try to capture some dynamics. A future model for the data probably cannot do better with modelling just on itself — we would need to try other variables to try to model the process based on external variables. We tried with the average minimum and maximum daily temperatures observed at the airport, but there were no shared dynamics between the temperatures and precipitation amounts.

IV. References

The data are from NOAA.

They are extracted from:

NOAA's climate data online search tool, focused on the daily summaries over the date ranges from January 1, 1980, to January 1, 2024. BWI was then chosen as a search term, and this data then needed to be downloaded from this tool. The website to start the search is:

https://www.ncei.noaa.gov/cdo-web/search

There is no direct page to this data, it has to be downloaded from a link that is emailed.