Chapter 10
Forecasting in the Presence of Outliers or Time-Varying Variance

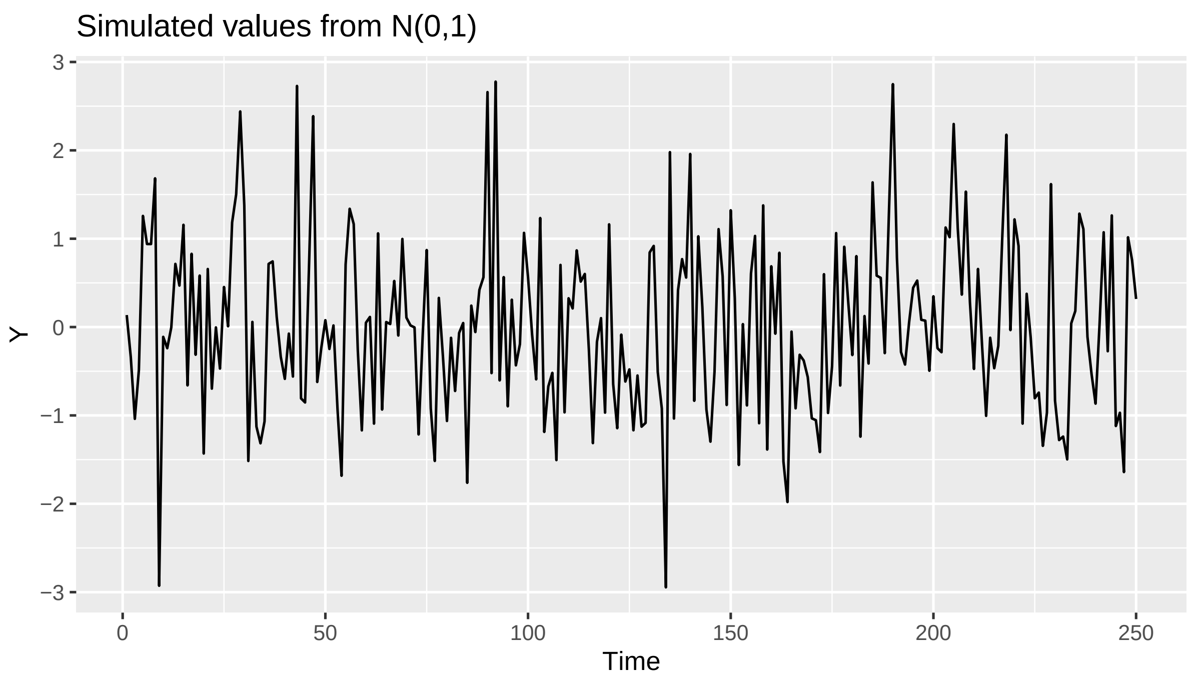
1 Introduction

We have now seen several methods of quantitative forecasting. While they all combine information in different ways to produce different forecasts, one thing that is common to all is that they assume normally distributed errors. Therefore, we can write any of the methods we have seen so far as:

$$y_t = \hat{y}_t + \varepsilon_t$$
$$\varepsilon_t \sim \mathcal{N}(0, \sigma)$$

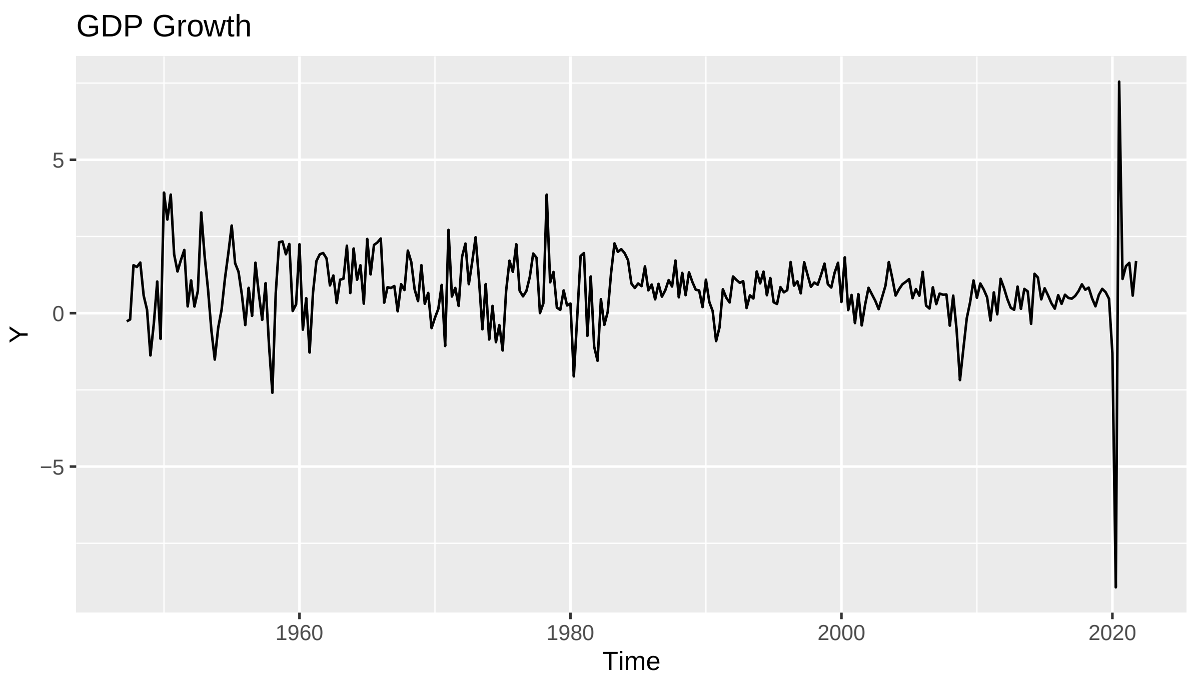
where $\hat{y}_t$ is the model fit for period t (i.e. it is the model's "best guess" of y_t).

Models that assume normally distributed error terms are popular because computation of model parameters is relatively simple when using squared error loss. However, a downside to assuming normally distributed error terms is that under a normal distribution, outlier values are highly improbable. For example, let's assume that the error terms are distributed $\mathcal{N}(0, 1)$ (i.e. mean=0 and SD=1). We can simulate values from this distribution:



Notice that of the 250 values simulated, none lie outside of the interval (-3,3). In fact, under a normal distribution, the probability that a value lies within +/-3 standard deviations from the mean is 99.7%. The probability that a value lies within +/-4 standard deviations is 99.99%. In other words, if a variable is truly normally distributed, then it is a virtual certainty that you would ever find it's value to be more than 4 standard deviations from its mean, unless you had collected a very large sample.

But why is this a problem? In economic data, especially after COVID-19, it is common to see values that are much more extreme than 4 standard deviations. For example, this is the plot of GDP growth:



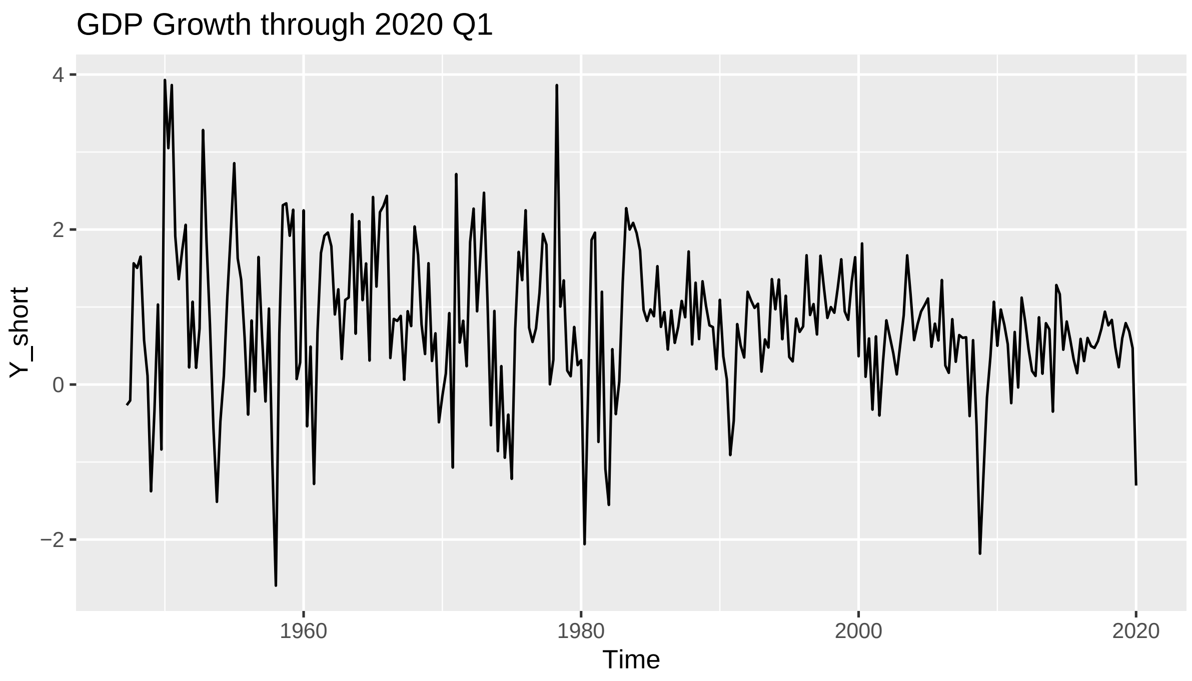
There are at least two clear outlier values. The smallest value, -8.94%, is more than 7 standard deviations from the mean. The largest value, 7.55%, is more than 6 standard deviations from the mean. Again, the normal distribution would suggest that values like these would be virtually impossible.

2 Practical Consequences of Outliers

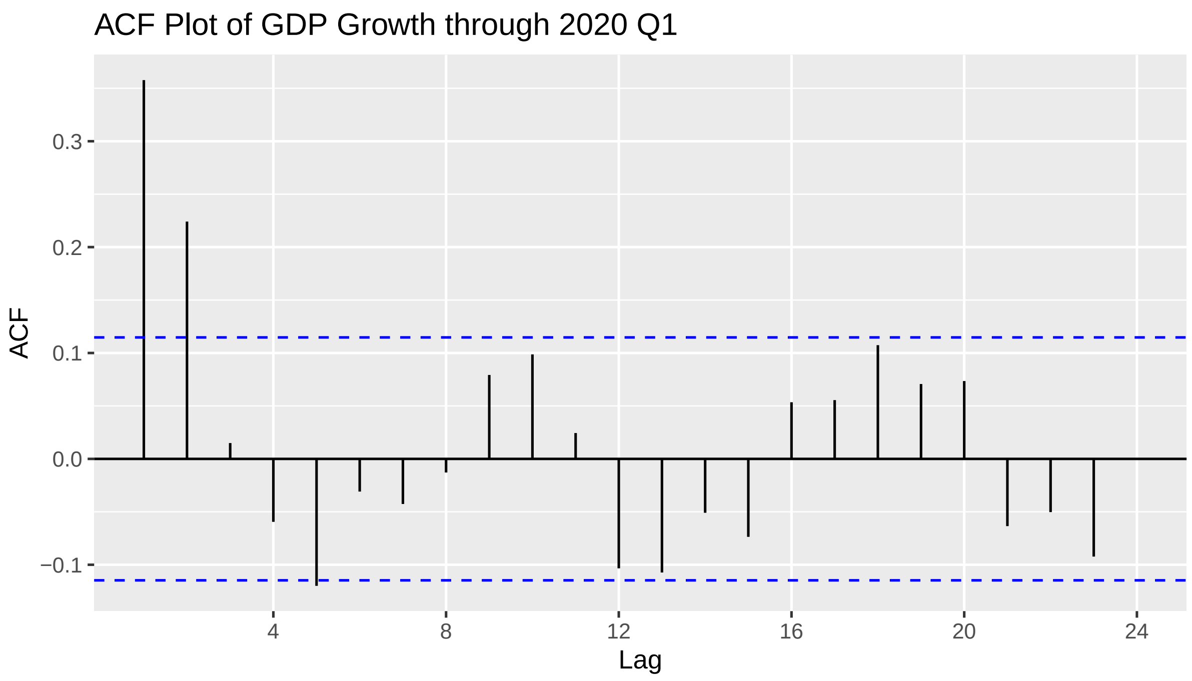
When extreme values are present in the data and errors are assumed to be normally distributed, the extreme values can have undue influence on parameter estimation. In other words, for most of the history of the data, when outliers are not present, there may be clear patterns that lead to accurate forecasts. However, even if the outliers are one or two period aberrations, with the data quickly returning to its normal historical patterns, the mere presence of the outliers can drastically alter parameter estimates, leading to worse forecasts in "normal" times.

We will use an ARIMA model fit to GDP growth as an illustrative example. We will assume an AR(2) model for GDP growth as is common in the economics literature. If we ignore the extreme values of GDP growth that occurred after 2020Q1, then we have the following plots and model estimates:

```
Y_short <- window(Y,end=c(2020,1))
autoplot(Y_short) + ggtitle("GDP Growth through 2020 Q1")
```



```
ggAcf(Y_short) + ggtitle("ACF Plot of GDP Growth through 2020 Q1")
```

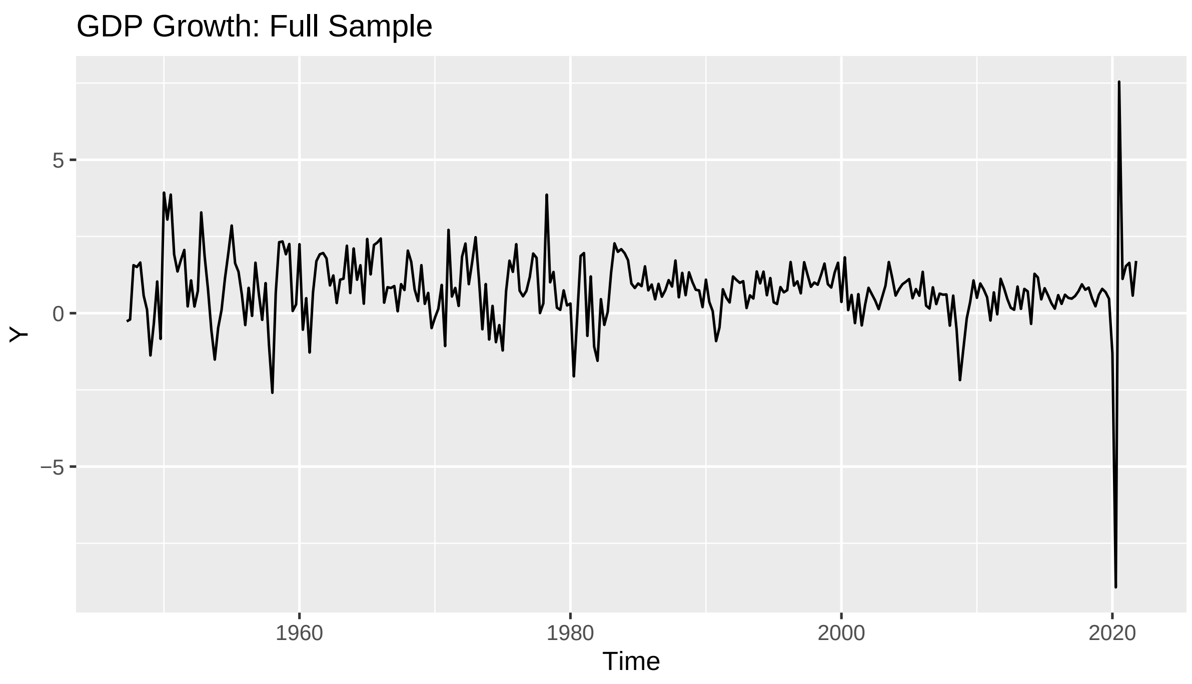


```
fit <- Arima(Y_short,order=c(2,0,0))
summary(fit)

## Series: Y_short
## ARIMA(2,0,0) with non-zero mean
##
## Coefficients:
##          ar1     ar2    mean
##       0.3238  0.1121  0.7625
## s.e.  0.0586  0.0586  0.0901
##
## sigma^2 estimated as 0.7664:  log likelihood=-374.06
## AIC=756.11   AICc=756.25   BIC=770.82
##
## Training set error measures:
##                       ME      RMSE       MAE       MPE     MAPE      MASE
## Training set 0.002328683 0.8709168 0.6323358 -194.8495 306.1794 0.6329029
##                     ACF1
## Training set 0.01240332
```

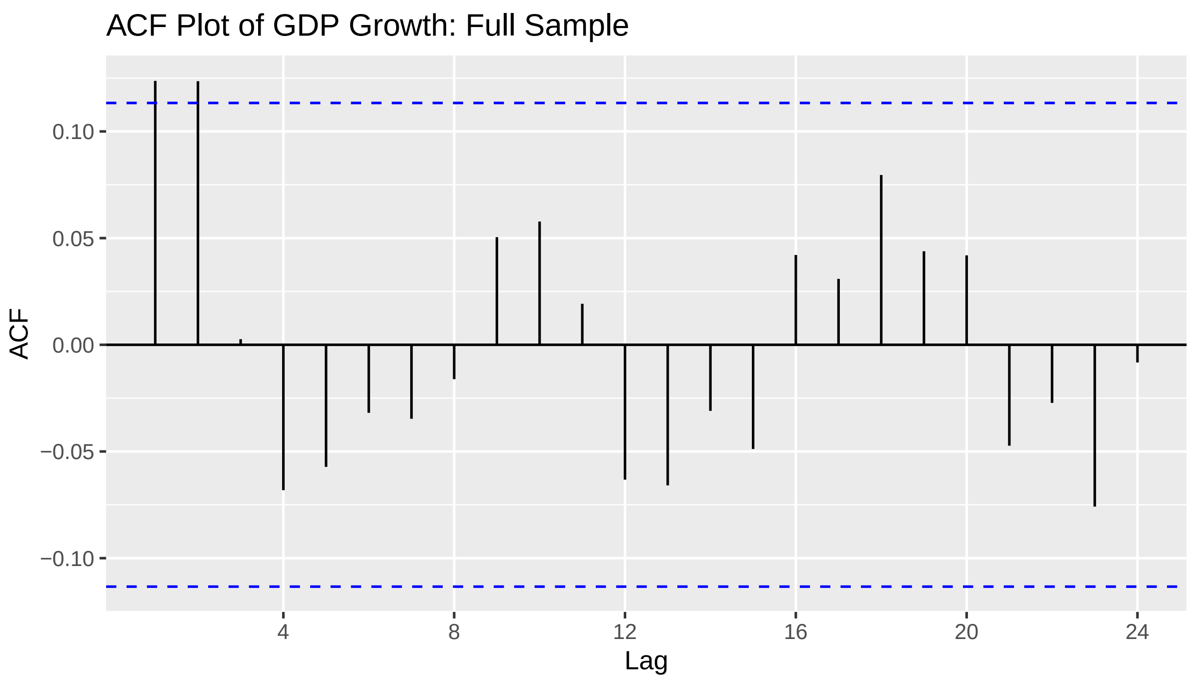
We can see that throughout these 73 years of GDP data, there is a mild but clear positive autocorrelation pattern. Quarters with above average GDP growth tend to be followed by a future quarter or two of above average GDP growth, and quarters with below average GDP growth tend to be followed by a future quarter or two of below average GDP growth. This is likely due to the "business cycle" — GDP growth tends to stay above average throughout expansions and then turns negative for a few quarters during recessions. Therefore, after estimating the AR(2) model we can see that both AR coefficients are estimated to be significantly above 0.

Now let's see what happens if we include the all historical data, including the economic disruption due to COVID-19:

```
autoplot(Y) + ggtitle("GDP Growth: Full Sample")
```

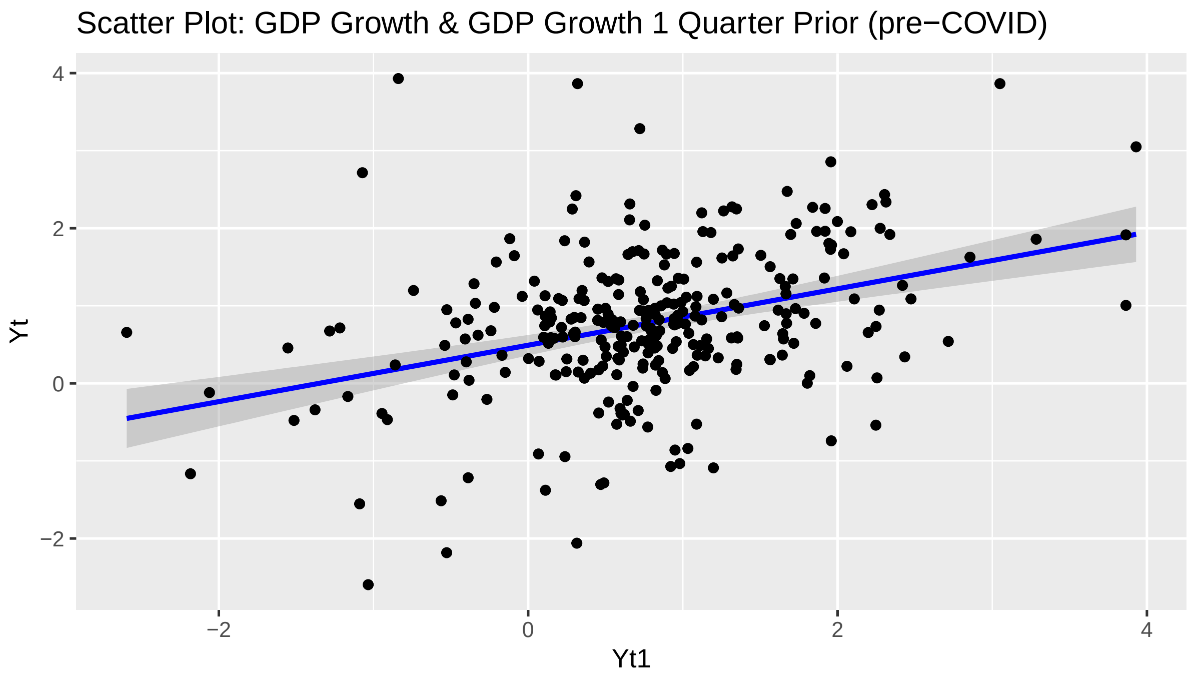


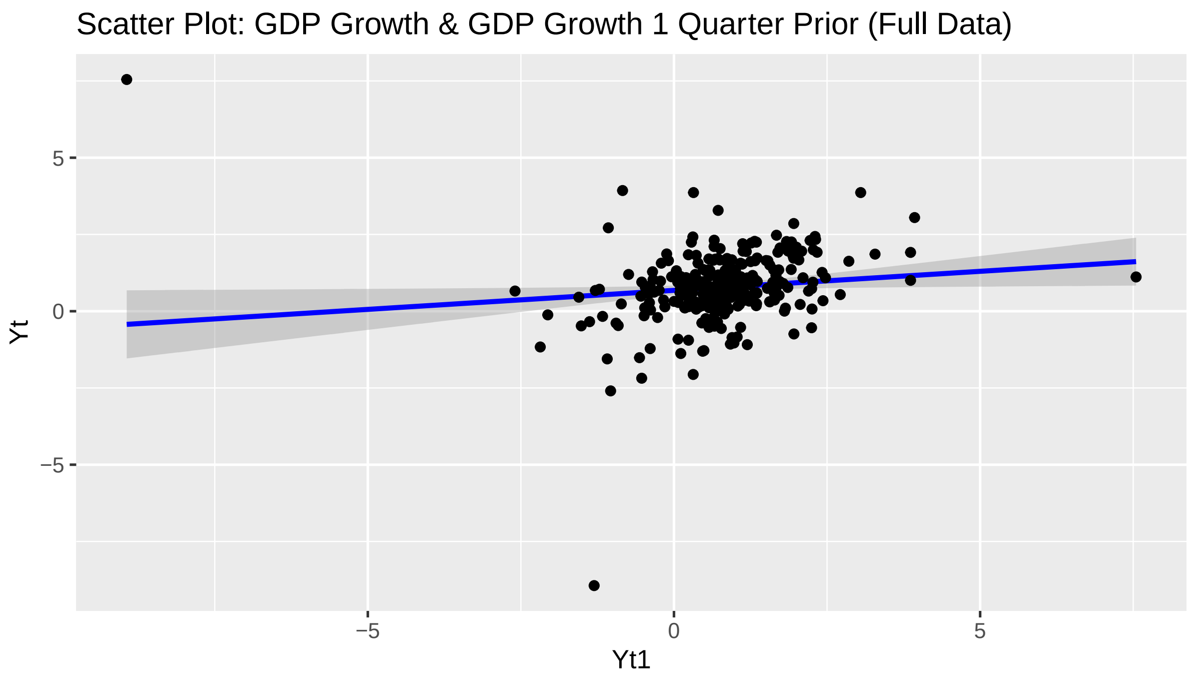
```r
ggAcf(Y) + ggtitle("ACF Plot of GDP Growth: Full Sample")
```



We can see that the inclusion of just these two quarters sharply decreases the estimated autocorrelation at the first lag. Intuitively, this is because we see a massive negative outlier followed by a massive positive outlier. The drastic flip from negative to positive pushes against the more typical pattern of positive following positive and negative following negative that we have seen over the previous 73 years. Since these values are so extreme, they are able to mute the much more typical pattern that has persisted over quite some time.

Another way we can see the undue impact of these outliers by looking at the scatter plot of the data in each quarter compared to the data in the quarter preceding it. The slope of the line of best fit is the autocorrelation coefficient at the first lag. We can see that after the inclusion of data during the COVID-19 pandemic, the slope of the line flattens substantially, due to the new outliers that appear at the very top left of the graph and at the far right of the graph. These points distort the historic correlation between GDP growth in each quarter and the next.





We can also see their impact on the estimated AR(2) parameters:

```
fit <- Arima(Y,order=c(2,0,0))
summary(fit)
```

```
## Series: Y
## ARIMA(2,0,0) with non-zero mean
##
## Coefficients:
##          ar1     ar2    mean
##       0.1104  0.1101  0.7702
## s.e.  0.0575  0.0575  0.0846
##
## sigma^2 estimated as 1.316:  log likelihood=-463.83
## AIC=935.67   AICc=935.8   BIC=950.47
##
## Training set error measures:
##                       ME     RMSE       MAE       MPE     MAPE      MASE
## Training set 0.001237699 1.141428 0.6976893 -152.9154 272.3546 0.6314224
##                    ACF1
## Training set 0.00262921
```

While both parameters are still positive, ϕ_1 is much closer to 0 and is barely statistically significant. Using the model with these parameters would result in different (and for most of the last 70+ years, worse) forecasts than the forecasts made with the model estimated on data that excluded the pandemic.

3 Solutions

We have seen that in the AR model with normally distributed errors, one or two periods of outlier data can overturn historical patterns built up over decades, almost certainly leading to worse forecast performance in future periods. There are several ways that we could address this problem:

1. Use a "fat-tailed" distribution (like a Student-t distribution) for the errors, instead of a Normal distribution. With a fat-tailed distribution, extreme values are assumed to happen more frequently and have less of an impact on parameter estimates.

2. Let the standard deviation of the Normal distribution change over time. In periods where the SD is higher, extreme values are more likely and therefore have less impact on the parameter estimates.

3. Use dummy variables for the outlier periods. The model would then be able to fit these periods nearly perfectly. Even though the observed values would still be extreme, they would not impact the parameter estimates.

4. Replace the extreme values with less extreme values. For example, any time a value is more than 4 standard deviations from the mean, you would replace that value with the value that is exactly 4 standard deviations from the mean. This would alleviate (but not eliminate) the impact of the outliers on the parameter estimates.

5. If the outlier values are at the very end of the series, you could do what we did above and ignore those values during estimation. However, this is only a temporary solution, since as time progresses the incoming values are likely to be more typical, and ignoring them would mean that the parameter estimates would never be updated again (e.g. in our example we would end estimation in 2020Q1 and never update these estimates again). Generally speaking this is not considered good practice.

While any of the strategies 1–4 are likely to greatly alleviate the problem, some are more difficult than others to implement when using standard software packages. For example, the Arima function in R does not allow for (1) non-normal errors or for (2) the standard deviation of the error terms to change over time. In addition, methods like ETS do not allow for independent variables, so the inclusion of dummies will not work for ETS models. Therefore, we will first explore strategy number 4 (replacing outlier values), as it is the easiest to implement across any possible model in R and will typically lead to similar results as the other strategies. A major downside of using this strategy (or strategy 3), relative to strategy 1 or 2, is that if we use replacement values (or dummy variables) our model will not be able to recognize that the data experienced a period of increased volatility. Therefore, the prediction intervals (especially those that immediately follow the outlier values) will be narrower than they should be.

To implement the fourth strategy – replacing outlier values – we will need to determine in which periods outliers occurred and also determine replacement values for those outliers. Luckily, there is a built in function in the forecasting package in R that allows us to do this relatively easily. Then, in order to form forecasts, we will fit our model(s) on this alternative data, and apply those model fits to the unadjusted data. To do this we will follow the following steps:

1. Identify periods in which outlier values occurred and identify non-outlier replacement values for those periods.

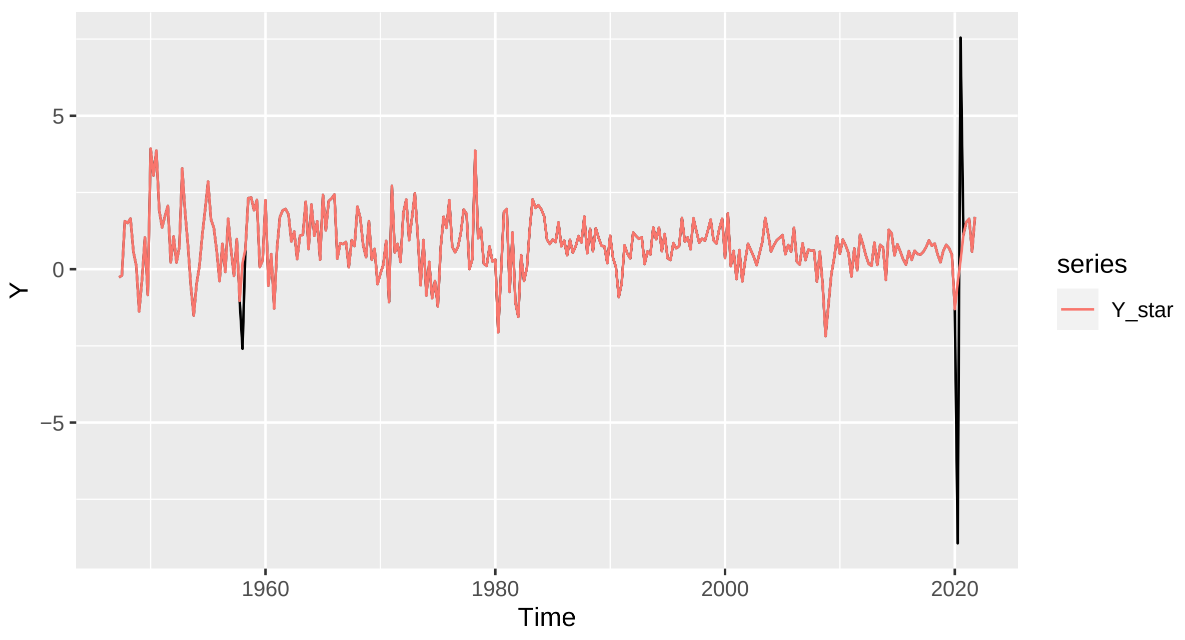
2. Replace the outlier values with the non-outlier replacement values to form an alternative data set, Y^*

3. Fit any of our forecasting models on the alternative data and save the parameter estimates.

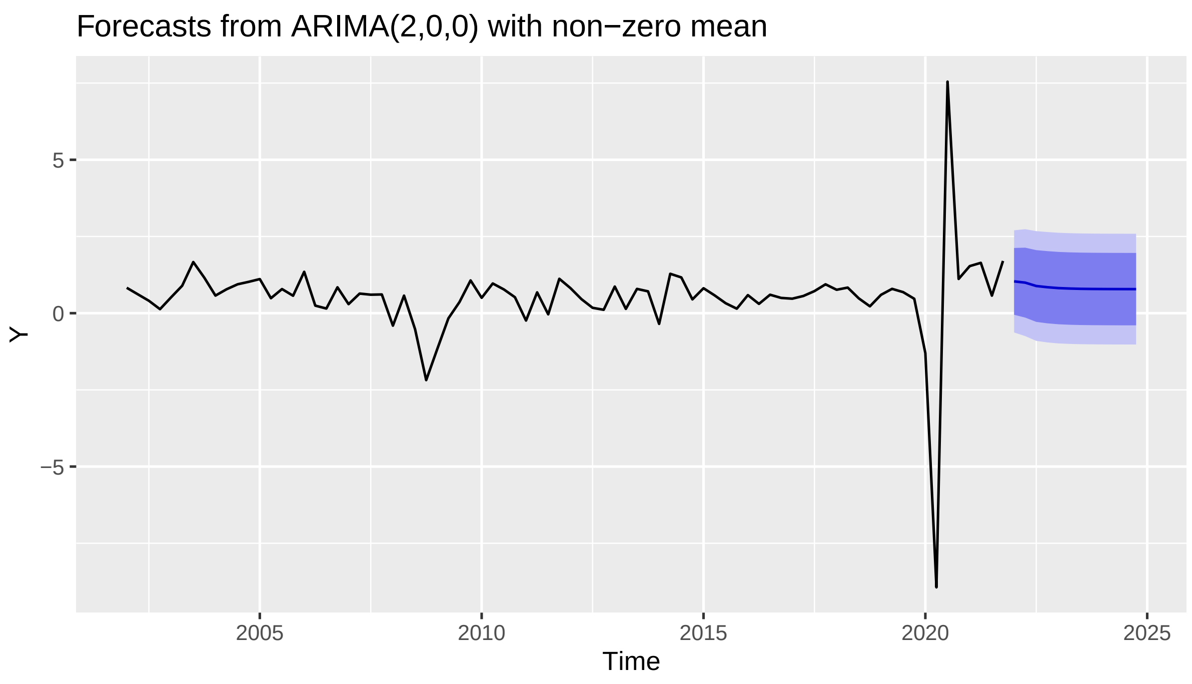
4. Apply the model with the parameter estimates from the previous step to the actual data, Y, to form forecasts.

Here is how this looks in R code. The model we use here is the AR(2) model, although we could use any we have learned in this course.

```
# Step 1: Identify outlier periods and replacement values using
# the tsoutliers() function
outliers <- tsoutliers(Y)
# Make a copy of the actual data called Y_star
Y_star <- Y

# Step 2: Create the alternate data set by replacing the actual
# outlier values with their replacement values
Y_star[outliers$index] <- outliers$replacements

# Plot the actual data and the alternative data
autoplot(Y) + autolayer(Y_star)
```



```
# Step 3: Fit the forecasting model to the alternative data
# In this example we will use an Arima model, however you could use
# any model (such as ETS or regression) in this step
fit_no_outlier <- Arima(Y_star,order=c(2,0,0))

# Step 4: Apply the estimated parameters to the actual data
# and forecast.
fit_unadj <- Arima(Y,model=fit_no_outlier)
fcst <- forecast(fit_unadj,h=12)
autoplot(fcst,include=80)
```



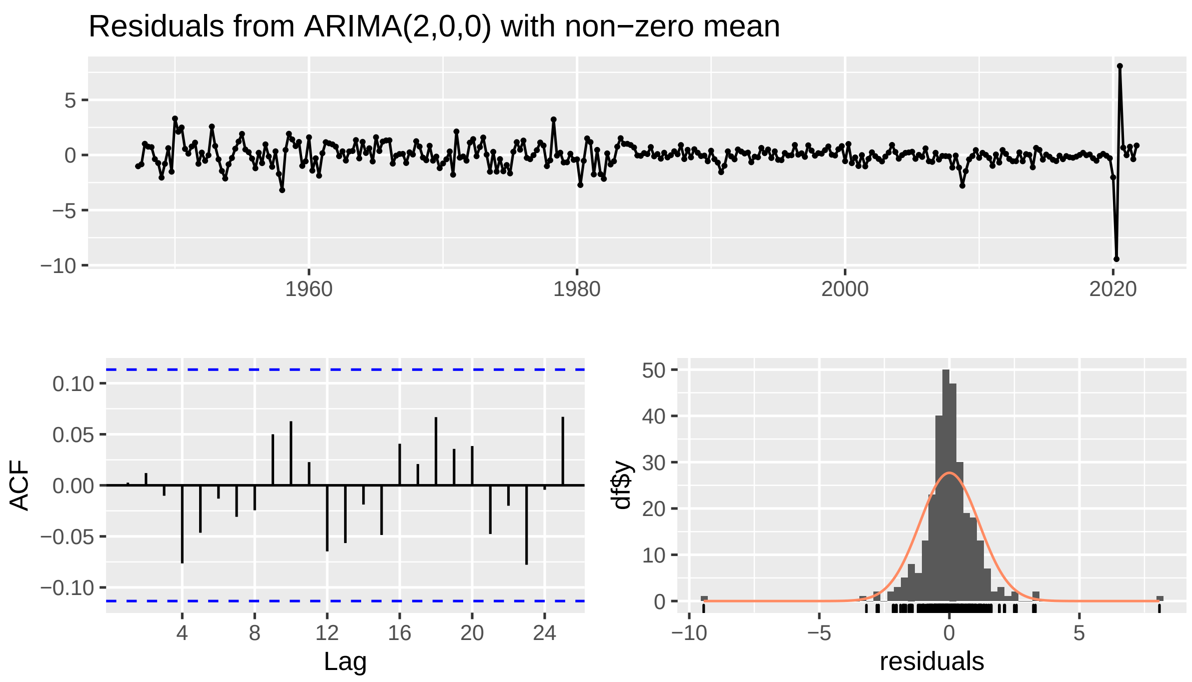
3.1 GARCH Errors

Another strategy (strategy 2 listed above) is to allow the standard deviation of the error term to change over time. This is common practice in fields like finance, where returns are unpredictable, but volatility is persistent. One major drawback of using GARCH models in R is that they will only work well with data that has been seasonally adjusted.[1]

For GDP growth data, we can fit an ARIMA model and inspect the residuals for evidence that the variance is changing over time. First, we fit an AR(2) model for GDP growth and check the residuals:

```
fit <- Arima(Y,order=c(2,0,0))
checkresiduals(fit)
```

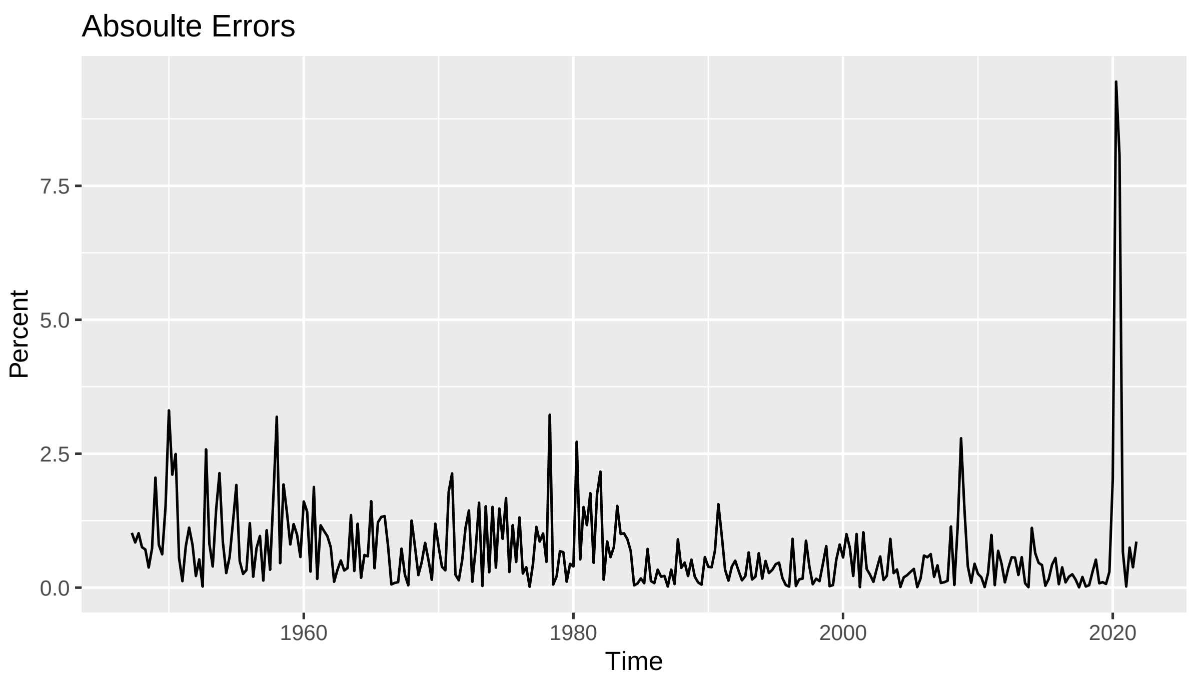
[1]In theory, you could use a seasonal ARIMA model for the data and apply GARCH errors. Unfortunately, in practice, the packages in R that estimate GARCH models force you to choose a non-seasonal ARIMA model. Hopefully this will eventually change.



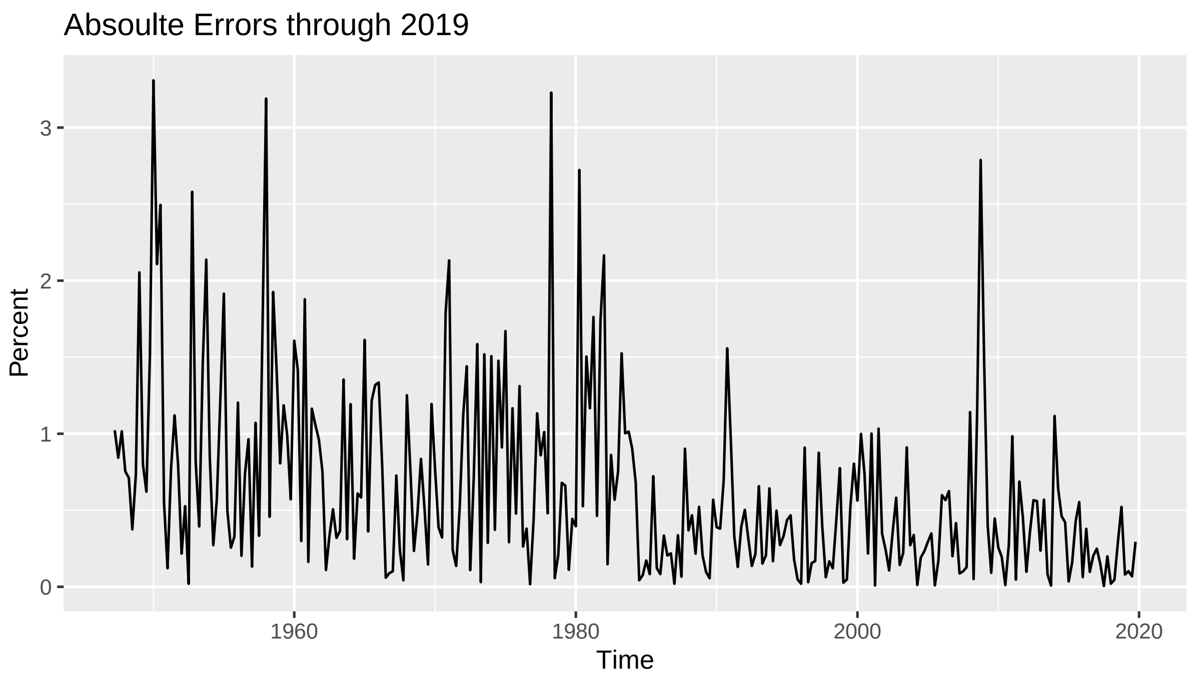
```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,0) with non-zero mean
## Q* = 3.0501, df = 5, p-value = 0.6923
##
## Model df: 3.   Total lags used: 8
```

In the time plot, we can see that it looks like the errors were more volatile before roughly 1980, and of course, became extremely volatile again during the COVID pandemic. We can see this more clearly if we plot the absolute value of the residuals, since we are interested in the size of the errors over time (and not necessarily if the errors were positive or negative). Since the COVID errors are so large, I will separate this into two plots. The first will cover the entire time series, while the second will end the data in 2019, to make any changes prior to COVID clearer.

```
errors <- resid(fit)
autoplot(abs(errors)) +
  ggtitle("Absoulte Errors") +
  ylab("Percent")
```



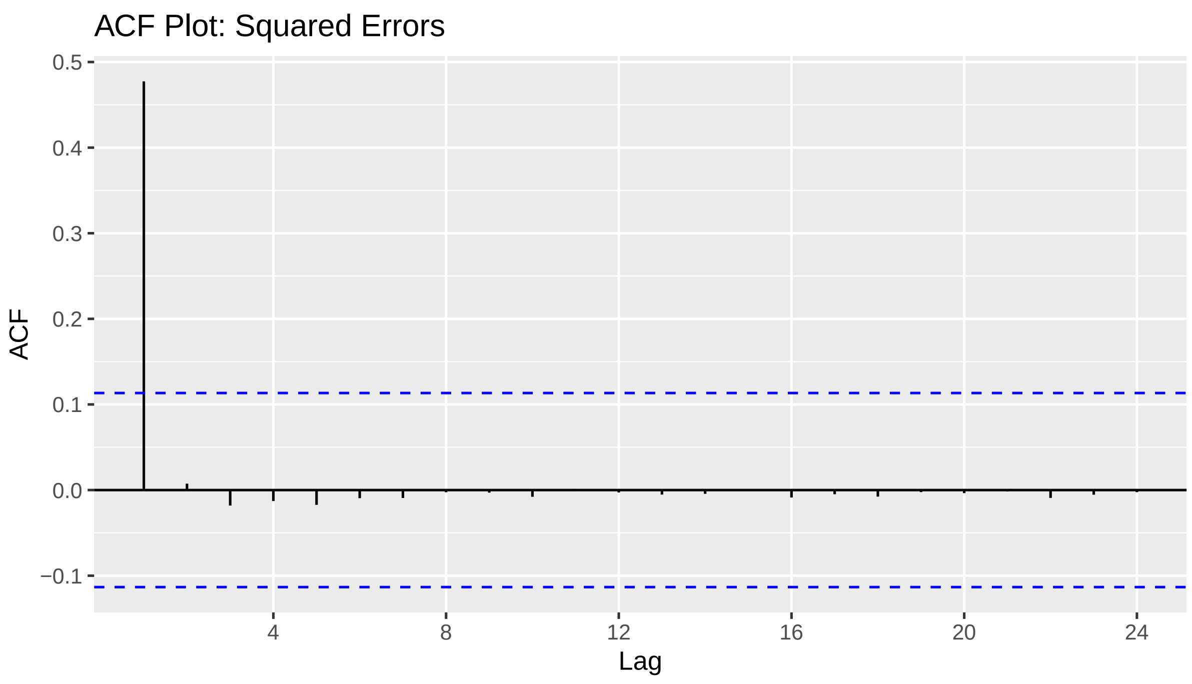
```r
errors_through_2019 <- window(errors,end=c(2019,4))
autoplot(abs(errors_through_2019)) +
  ggtitle("Absoulte Errors through 2019") +
  ylab("Percent")
```



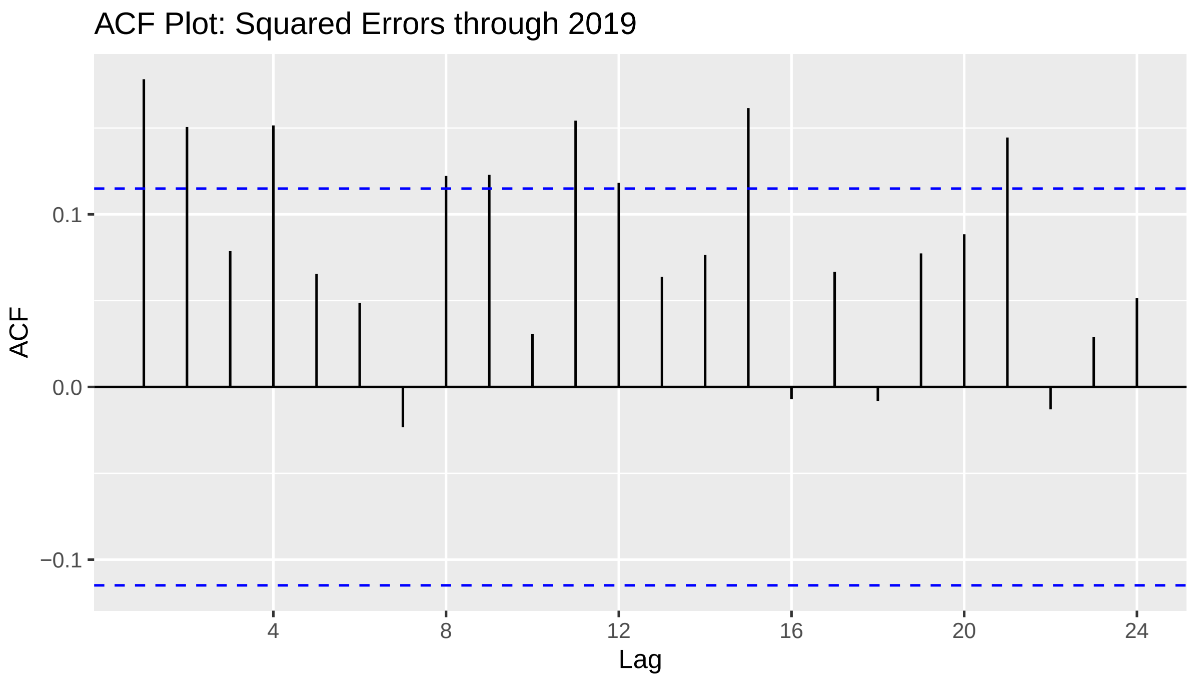
This confirms what we saw in the time plot of the residuals, with the average size of the residual being larger prior to roughly 1980 and again during COVID (and the Great Recession).

Since we will be modeling the variance, we can also look at the ACF plot of the squared errors:[2]

```
ggAcf(errors^2) +
  ggtitle("ACF Plot: Squared Errors")
```



```
ggAcf(errors_through_2019^2) +
  ggtitle("ACF Plot: Squared Errors through 2019")
```



[2]We used absolute errors on the time plot, since once we square the errors, the two large COVID errors get so big that it is hard to see the other data.

While the exact pattern of autocorrelation depends on when we end the data, it is clear that the squared errors have some positive autocorrelation. In other words, the size of past errors is predictive of the size of future errors; when past errors were large, future errors will tend to be large, and vice-versa.

To model this, we can use a GARCH(p,q) model. GARCH stands for Generalized Autoregressive Conditional Heteroscedasticity. Essentially, it just means that the variance can be modeled as an autoregressive process. In our case, we will model the data as follwing an ARMA(2,0) process, with GARCH(p,q) errors. The model would be written as:

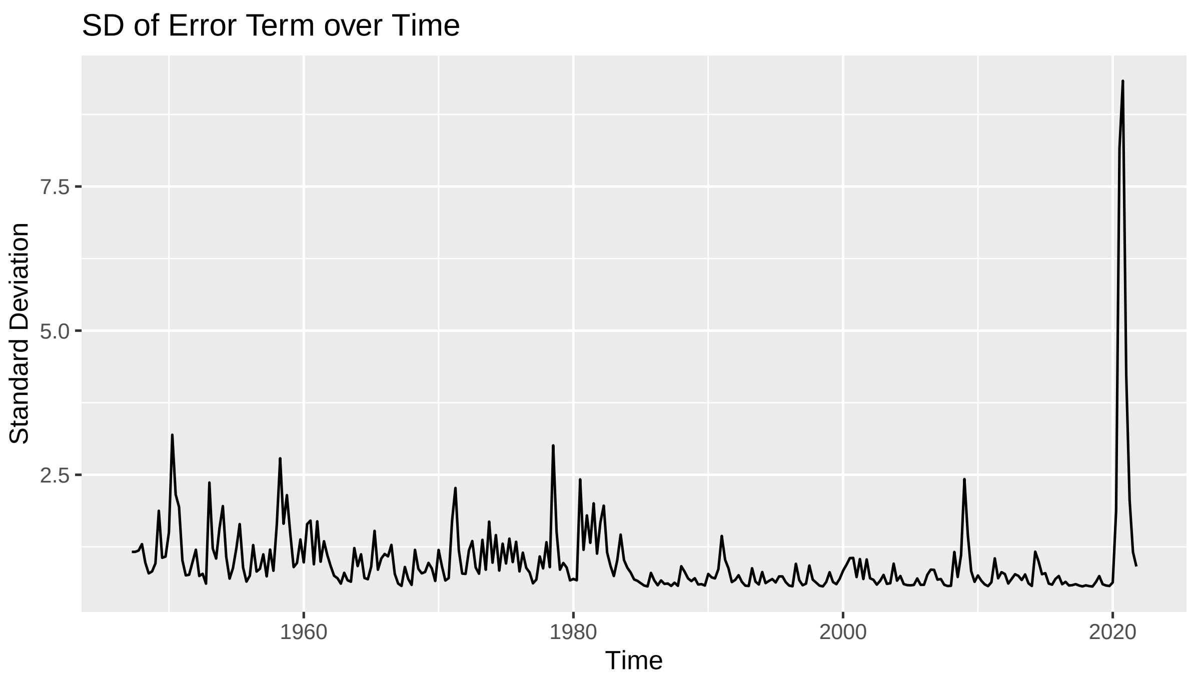
$$
\begin{aligned}
&y_t = b + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \varepsilon_t \\
&\varepsilon_t \sim \mathcal{N}(0, \sigma_t) \\
&\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \cdots + \alpha_q \varepsilon_{t-q}^2 + \beta_1 \sigma_{t-1}^2 + \cdots + \beta_p \sigma_{t-p}^2
\end{aligned}
$$

We can see that the variance of the error term can now vary over time. It depends on the "q" most recent residuals, and can also be persistent, so that it also depends on the "p" most recent lags of itself. Typically in GARCH models, $p = 1$. A very common GARCH model is the GARCH(1,1) model. If we still assume an ARMA(2,0) process of the data with GARCH(1,1) errors, we would have:

$$
\begin{aligned}
&y_t = b + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \varepsilon_t \\
&\varepsilon_t \sim \mathcal{N}(0, \sigma_t) \\
&\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2
\end{aligned}
$$

To fit a GARCH model, you will need to download, install, and load the "rugarch" package. Then you could fit the model described above (ARMA(2,0) with GARCH(1,1) errors):

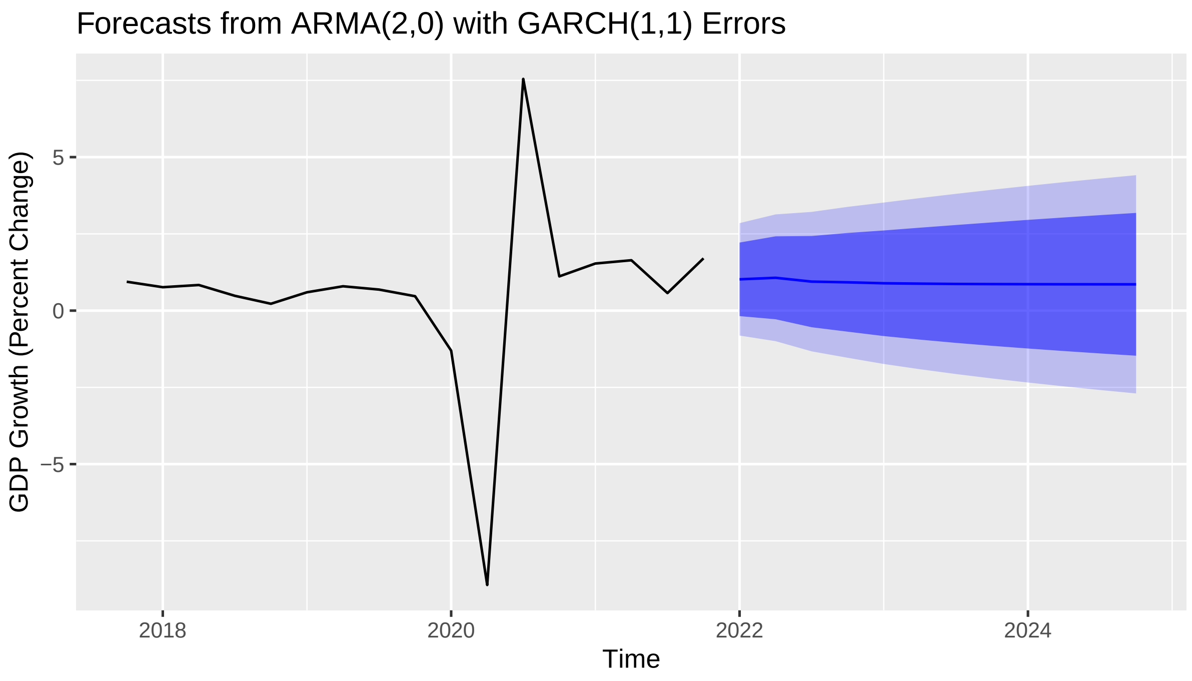
```r
library(rugarch)
# Specify model
spec <- ugarchspec(variance.model = list(model="sGARCH",
                                         garchOrder= c(1,1)),
                   mean.model = list(armaOrder=c(2,0)),
                   distribution.model="norm")
  # Fit model
  garch <- ugarchfit(spec=spec,
                     data=Y,
                     solver="hybrid",
                     solver.control=list(trace=1))

##
## Iter: 1 fn: 389.7998  Pars:  0.85571 0.26073 0.20170 0.25030 0.78424 0.20047
## Iter: 2 fn: 389.7998  Pars:  0.85571 0.26073 0.20170 0.25030 0.78424 0.20047
## solnp--> Completed in 2 iterations

  # Print AIC of model
  print(infocriteria(garch)*T)

##
## Akaike       791.5995
## Bayes        813.8022
## Shibata      791.3650
## Hannan-Quinn 800.4860

  # Print AIC of ARMA(2,0) with constant variance as comparison
  print(AIC(fit_unadj))

## [1] 942.4718

  # Print Estimated Coefficients
  print(garch@fit$coef)

##        mu       ar1       ar2     omega    alpha1     beta1
## 0.8557144 0.2607292 0.2016958 0.2502970 0.7842444 0.2004666

  # Plot estimated standard deviation over time
  sd_GARCH <- ts(garch@fit$sigma,start=c(1947,2),frequency = 4)
  autoplot(sd_GARCH) +
    ggtitle("SD of Error Term over Time") +
    ylab("Standard Deviation")
```



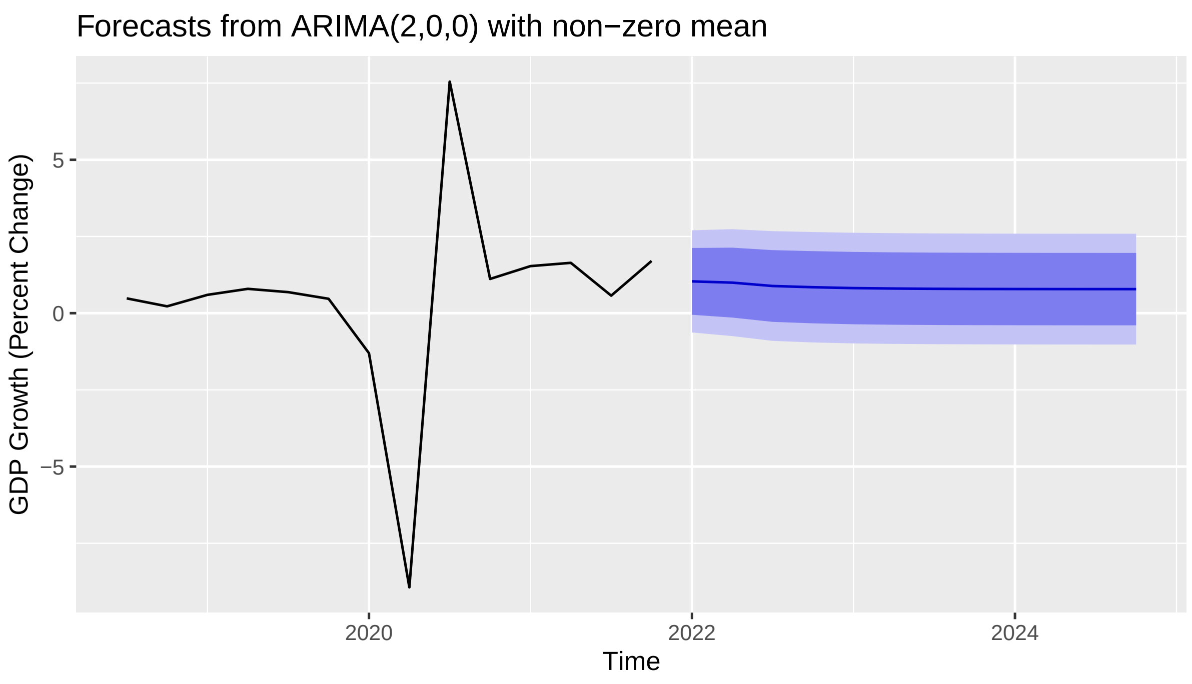
We can also create forecasts from the GARCH model. This takes a bit more work then we are used to, since this model is not part of the forecast package we are used to using.

```
# Get forecast
garch_forecast <- ugarchforecast(garch,n.ahead=12,data=Y)
# Pull out point-forecast
Y_hat <- garch_forecast@forecast$seriesFor
# Declare as time-series
Y_hat <- ts(Y_hat,start=c(year,(quarter+1)),frequency = 4)
# Create 80 and 95% prediction intervals using the forecasted SD
# of the error term
Y_lower95 <- ts(Y_hat - 1.96*garch_forecast@forecast$sigmaFor,
                start=c(year,(quarter+1)),frequency = 4)
Y_upper95 <- ts(Y_hat + 1.96*garch_forecast@forecast$sigmaFor,
                start=c(year,(quarter+1)),frequency = 4)
Y_lower80 <- ts(Y_hat - 1.282*garch_forecast@forecast$sigmaFor,
                start=c(year,(quarter+1)),frequency = 4)
```

```r
Y_upper80 <- ts(Y_hat + 1.282*garch_forecast@forecast$sigmaFor,
                start=c(year,(quarter+1)),frequency = 4)

# Plot forecasts
autoplot(Y_hat,color="blue") +
  geom_ribbon(aes(ymin=Y_lower80,ymax=Y_upper80),alpha=0.5,fill="blue") +
  geom_ribbon(aes(ymin=Y_lower95,ymax=Y_upper95),alpha=0.2,fill="blue") +
  autolayer(window(Y,start=c(2017,4)),color="black") +
ggtitle("Forecasts from ARMA(2,0) with GARCH(1,1) Errors") +
ylab("GDP Growth (Percent Change)")
```

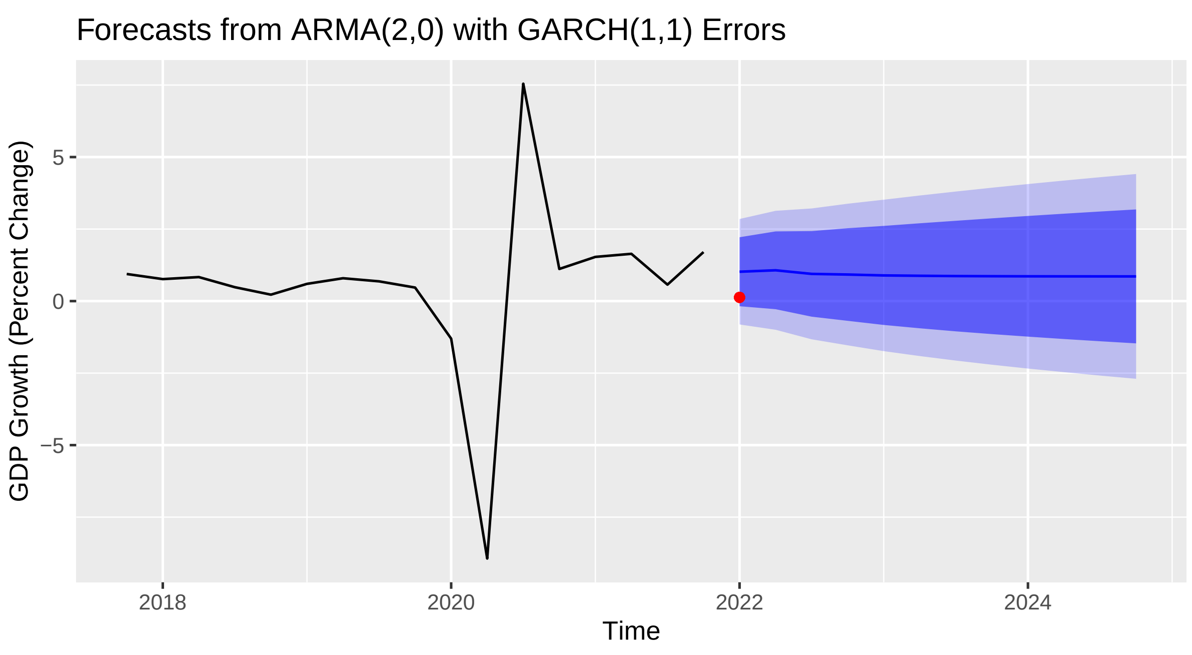


```r
# Compare to forecasts without GARCH errors
fit_unadj <- Arima(Y,model=fit_no_outlier)
fcst <- forecast(fit_unadj,h=12)
df <- data.frame(t=(year+(quarter)/4),Y=2.64)
autoplot(fcst,include=14) +
  ylab("GDP Growth (Percent Change)")
```

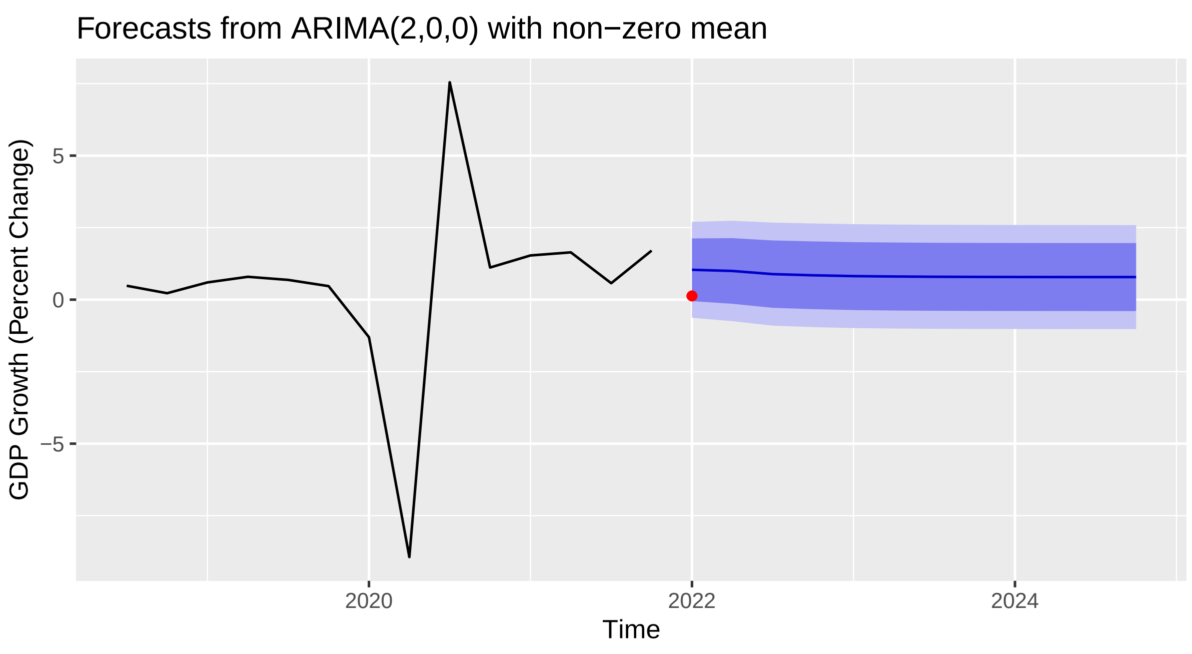


Finally, we can repeat these graphs, overlaying the RGDP growth forecast from the Atlanta Fed's "GDP Now" forecasting model as a red dot. This forecasting model is highly accurate, and their last forecast for the quarter in question usually lands within $+/-0.25\%$ of actual RGDP growth. Therefore, the red dot represents a more plausible likely value for RGDP growth than the forecast from our AR(2) model.

```r
data <- fredr(series_id = "GDPNOW")
T <- length(data$value)
Yf <- data$value[T]
Yf <- 100*((1+Yf/100)^(1/4)-1)

# Plot forecasts with GDPNow Forecast
autoplot(Y_hat,color="blue") +
  geom_ribbon(aes(ymin=Y_lower80,ymax=Y_upper80),alpha=0.5,fill="blue") +
  geom_ribbon(aes(ymin=Y_lower95,ymax=Y_upper95),alpha=0.2,fill="blue") +
  autolayer(window(Y,start=c(2017,4)),color="black") +
  geom_point(x=(year+quarter/4),y=Yf,color="red") +
ggtitle("Forecasts from ARMA(2,0) with GARCH(1,1) Errors") +
ylab("GDP Growth (Percent Change)")
```



```r
# Compare to forecasts without GARCH errors
fit_unadj <- Arima(Y,model=fit_no_outlier)
fcst <- forecast(fit_unadj,h=12)
autoplot(fcst,include=14) +
  geom_point(aes(x=c(year+quarter/4),y=c(Yf)),color="red") +
ylab("GDP Growth (Percent Change)")
```



3.2 Stochastic Volatility (SV)

Stochastic volatility (SV) is another way to capture changing variance over time. It is a similar solution to using GARCH errors, but it is a slightly different model. Whereas a GARCH model assumes that the variance depends on past errors and past values of itself, the SV model assumes the variance depends on past values of itself as well as shocks that directly impact the variance, but not the data. In many finance and economics applications SV models outperform GARCH models in forecast accuracy, and they have increased in use substantially over the past 20 years.[3]

Like the GARCH model, we need to assume an ARMA process for the data. We will continue to assume that the data is best described by an AR(2) model. Then, we would write the SV model, which typically assumes the log of the variance evolves according to an AR(1) model, as follows:

$$\begin{aligned} y_t &= b + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \varepsilon_t \\ \varepsilon_t &\sim \mathcal{N}(0, \sigma_{\varepsilon,t}) \\ \sigma_{\varepsilon,t} &= \exp\left\{\frac{h_t}{2}\right\} \\ h_t &= \mu + \Phi h_{t-1} + v_t \\ v_t &\sim \mathcal{N}(0, \sigma_v) \end{aligned}$$

It is common to assume that h_t evolves according to a random walk, in which case we could simplify the equation for h_t

$$h_t = h_{t-1} + v_t$$

We will estimate this model in R, using the GDP growth data. To do so, you will need to download, install, and load the "stochvol" package.

```
library(stochvol)


# Fit the stochastic volatility model, forcing h_t to evolve according to
# a random walk
```

[3]Like GARCH models, the package estimating SV models does not allow for seasonal models, so you will need to use data that has been seasonally adjusted.

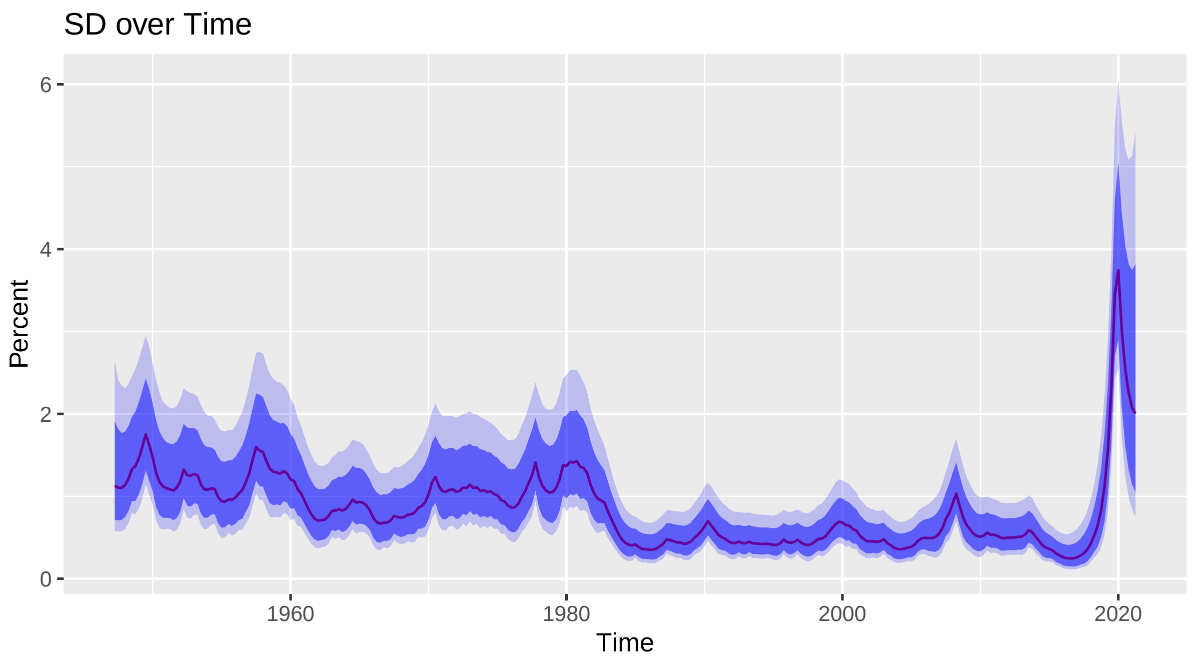
```
res_sv <- svsample(Y, # Insert data, in this case RGDP growth
                   designmatrix = "ar2", # Model for the data
                   draws=50000, # Number of simulations
                   burnin=10000, # Number of burn-in simulations (to throw away)
                   priorphi = c(10000,1), # Force Phi=1 for RW
                   priormu = c(0,0.0000001)) # Force mu=0 for RW

# Print a summary of the model
print(summary(res_sv))
```

```
##
## Summary of 'svdraws' object
##
## Prior distributions:
## mu        ~ Normal(mean = 0, sd = 1e-07)
## (phi+1)/2 ~ Beta(a = 10000, b = 1)
## sigma^2   ~ Gamma(shape = 0.5, rate = 0.5)
## nu        ~ Infinity
## rho       ~ Constant(value = 0)
## beta      ~ MultivariateNormal(...)
##
## Stored 50000 MCMC draws after a burn-in of 10000.
## No thinning.
##
## Posterior draws of SV parameters (thinning = 1):
##                mean      sd       5%      50%     95%   ESS
## mu        -3.9e-10 1.0e-07 -1.6e-07 -4.2e-10 1.6e-07 50000
## phi        1.0e+00 2.2e-04  1.0e+00  1.0e+00 1.0e+00    30
## sigma      4.5e-01 7.9e-02  3.3e-01  4.5e-01 5.9e-01  1921
## exp(mu/2)  1.0e+00 5.0e-08  1.0e+00  1.0e+00 1.0e+00 50000
## sigma^2    2.1e-01 7.4e-02  1.1e-01  2.0e-01 3.5e-01  1921
##
```

```
## Posterior draws of regression coefficients (thinning = 1):
##         mean    sd    5%  50%  95%   ESS
## beta_0 0.43 0.064 0.323 0.43 0.53 20686
## beta_1 0.25 0.062 0.148 0.25 0.35 22805
## beta_2 0.18 0.058 0.088 0.18 0.28 20968
```

We can plot the in-sample estimates of $\sigma_{\varepsilon,t}$, the SD of the error in the AR(2) model as it changes over time.

```
# Save the 2.5th, 10th, 50th (median), 90th, and 97.5th percentile values of
# h_t at each in-sample time-period.
hquant <- apply(res_sv$latent[[1]], 2, quantile, c(0.025,.1, .5, .9,0.975))
# Convert the value of h_t to values of sigma_t (the SD of the error)
sigmaquant <- apply(exp(res_sv$latent[[1]]/2), 2,
                    quantile, c(0.025,.1, .5, .9,0.975))

# Define lower and upper 80th percentile range for sigma (in-sample)
sigma_lower80 <- ts(sigmaquant["10%",],start=c(1947,2),frequency = 4)
sigma_upper80 <- ts(sigmaquant["90%",],start=c(1947,2),frequency = 4)
# Define lower and upper 90th percentile range for sigma (in-sample)
sigma_lower95 <- ts(sigmaquant["2.5%",],start=c(1947,2),frequency = 4)
sigma_upper95 <- ts(sigmaquant["97.5%",],start=c(1947,2),frequency = 4)
# Define median value of sigma
sigma_med <- ts(sigmaquant["50%",],start=c(1947,2),frequency = 4)

# Plot the estimated SD of the residuals (sigma) over time
autoplot(sigma_med,color="red") +
geom_ribbon(aes(ymin=sigma_lower80,ymax=sigma_upper80),alpha=0.5,fill="blue") +
geom_ribbon(aes(ymin=sigma_lower95,ymax=sigma_upper95),alpha=0.2,fill="blue") +
ggtitle("SD over Time") +
ylab("Percent")
```

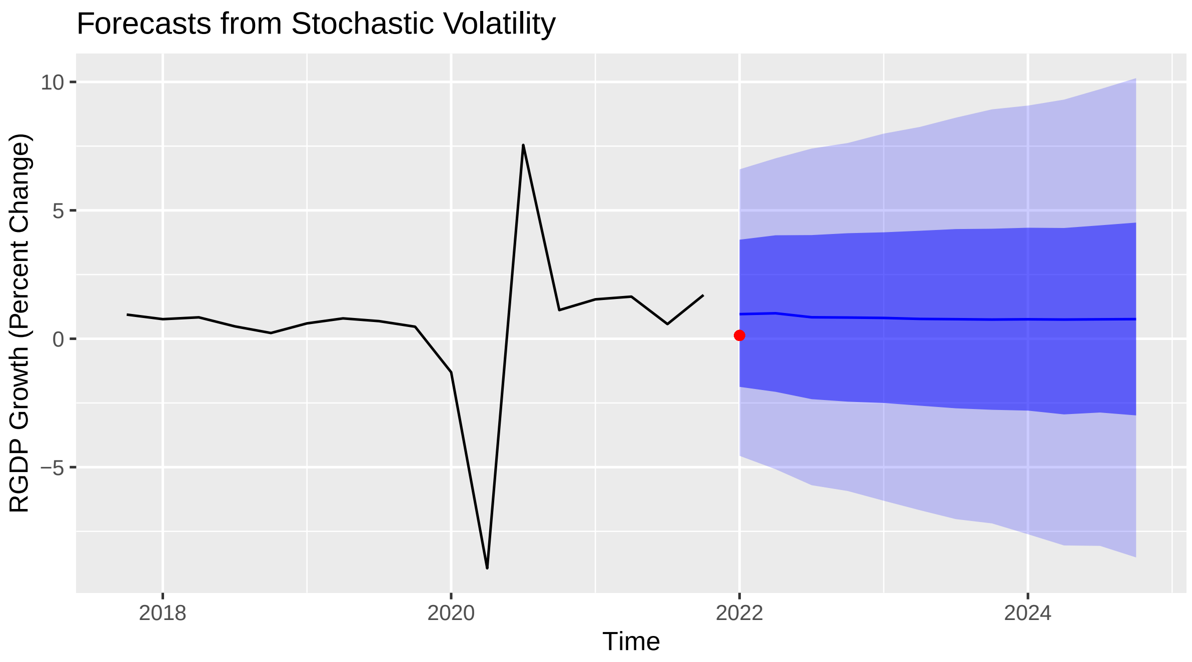


This is consistent with the absolute value of the error terms from the plain AR(2) model – the SD appears higher before about 1980, then falls to a lower level until the COVID pandemic, at which point it rises to about 4% per quarter. It then declines slowly to about 1.6% in 2021Q4.

After fitting the model, we need to create forecasts "by hand" since the forecast function will not work on SV models.

```
# Forecast 12 quarters ahead
ahead <- 12
# Create predictions from each simulation
preddraws <- predict(res_sv, steps = ahead)
# Save the 2.5th, 10th, 50th (median), 90th, and 97.5th percentile predictions
# for each forecast horizon
predquants <- apply(preddraws$y[[1]], 2, quantile, c(0.025,.1, .5, .9,0.975))
# Save lower and upper 80th percentile forecasts
Y_lower80 <- ts(predquants["10%",],start=c(year,(quarter+1)),frequency = 4)
Y_upper80 <- ts(predquants["90%",],start=c(year,(quarter+1)),frequency = 4)
# Save lower and upper 95th percentile forecasts
Y_lower95 <- ts(predquants["2.5%",],start=c(year,(quarter+1)),frequency = 4)
Y_upper95 <- ts(predquants["97.5%",],start=c(year,(quarter+1)),frequency = 4)
```

```r
# Save median forecast
Y_med <- ts(predquants["50%",],start=c(year,(quarter+1)),frequency = 4)

# Plot median forecasts with 80th and 95th percentile forecast intervals
# with GDPNOW forecast overlayed
autoplot(Y_med,color="blue") +
  geom_ribbon(aes(ymin=Y_lower80,ymax=Y_upper80),alpha=0.5,fill="blue") +
  geom_ribbon(aes(ymin=Y_lower95,ymax=Y_upper95),alpha=0.2,fill="blue") +
  autolayer(window(Y,start=c(2017,4)),color="black") +
  geom_point(aes(y=Yf,x=(year+(quarter/4))),color="red") +
  ggtitle("Forecasts from Stochastic Volatility") +
  ylab("RGDP Growth (Percent Change)")
```



Compared to the forecasts from the outlier-adjusted data from the GARCH model, the SV forecast has much wider forecast intervals. The GDPNOW forecast is well-within the 80% forecast interval from the SV model, while it was much closer to the boundary of the 80% interval produced by the other models.