Risk Forecasting Using ARMA-GARCH Models and ML: Value at Risk for the S&P 500 Index

2024-06-14

1. INTRODUCTION

The S&P 500 index is regarded as one of the most popular benchmarks for the U.S. stock market. It represents the performance of 500 leading publicly traded companies among various industries.

This project focuses on forecasting the volatility and risk metrics of the S&P 500 index using ARMA-GARCH models. We aim to capture the time-varying volatility through a Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model and use an ARMA model to account for the linear dependencies in the time series' mean.

Model selection is performed based on information criteria, prioritizing simplicity and stability. Using data from 2000 to 2024, we further assess the Value at Risk (VaR) and Expected Shortfall (ES), offering critical insights for financial risk management.

(Source: https://finance.yahoo.com/quote/%5EGSPC/)

2. METHOD

The project incurs the following procedure:

- 2.1. Data cleaning
- 2.2 Exploratory analysis.
- 2.3. Choice of methods.
- 2.4 Selection of ARMA model to capture the linear dependencies in the mean of the time series.
- 2.5 Sequential modeling and selection of GARCH model.

Importing necessary packages

```
library(MASS)
library(rugarch)
library(quantmod)
library(graphics)
library(xts)
library(timetk)
library(stats)
library(ggplot2)
library(forecast)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2     v tidyr     1.3.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::first()  masks xts::first(), rmgarch::first()
## x dplyr::lag()    masks stats::lag()
## x dplyr::last()   masks xts::last(), rmgarch::last()
## x purrr::reduce() masks rugarch::reduce()
## x dplyr::select() masks MASS::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

2.1 Data Cleaning

Importing the data

Source: https://finance.yahoo.com/quote/%5EGSPC/

```
GSPC <- quantmod::getSymbols("^GSPC", src="yahoo", return.class="xts",
from="2000-01-01", to="2024-01-01", auto.assign=F)

SP500 <- GSPC$GSPC.Close
#SP500 %>% glimpse

colnames(SP500) <- "SP"

SP500_r <- na.omit(diff(log(SP500))) * 100


SP500_r %>% glimpse
```

```
## An xts object on 2000-01-04 / 2023-12-29 containing:
##   Data:    double [6036, 1]
##   Columns: SP
##   Index:   Date [6036] (TZ: "UTC")
##   xts Attributes:
##     $ src      : chr "yahoo"
##     $ updated  : POSIXct[1:1], format: "2024-09-18 12:39:18"
##     $ na.action: 'omit' int 1
##   ..- attr(*, "index")= num 9.47e+08
```

```
SP500_r %>% head
```

```
##                     SP
## 2000-01-04 -3.90991753
## 2000-01-05  0.19203380
## 2000-01-06  0.09552217
## 2000-01-07  2.67299497
## 2000-01-10  1.11278249
## 2000-01-11 -1.31485768
```

```
SP500_r %>% tail
```

```
##                     SP
## 2023-12-21  1.02487693
## 2023-12-22  0.16586822
## 2023-12-26  0.42227610
## 2023-12-27  0.14294356
## 2023-12-28  0.03701061
## 2023-12-29 -0.28304770
```

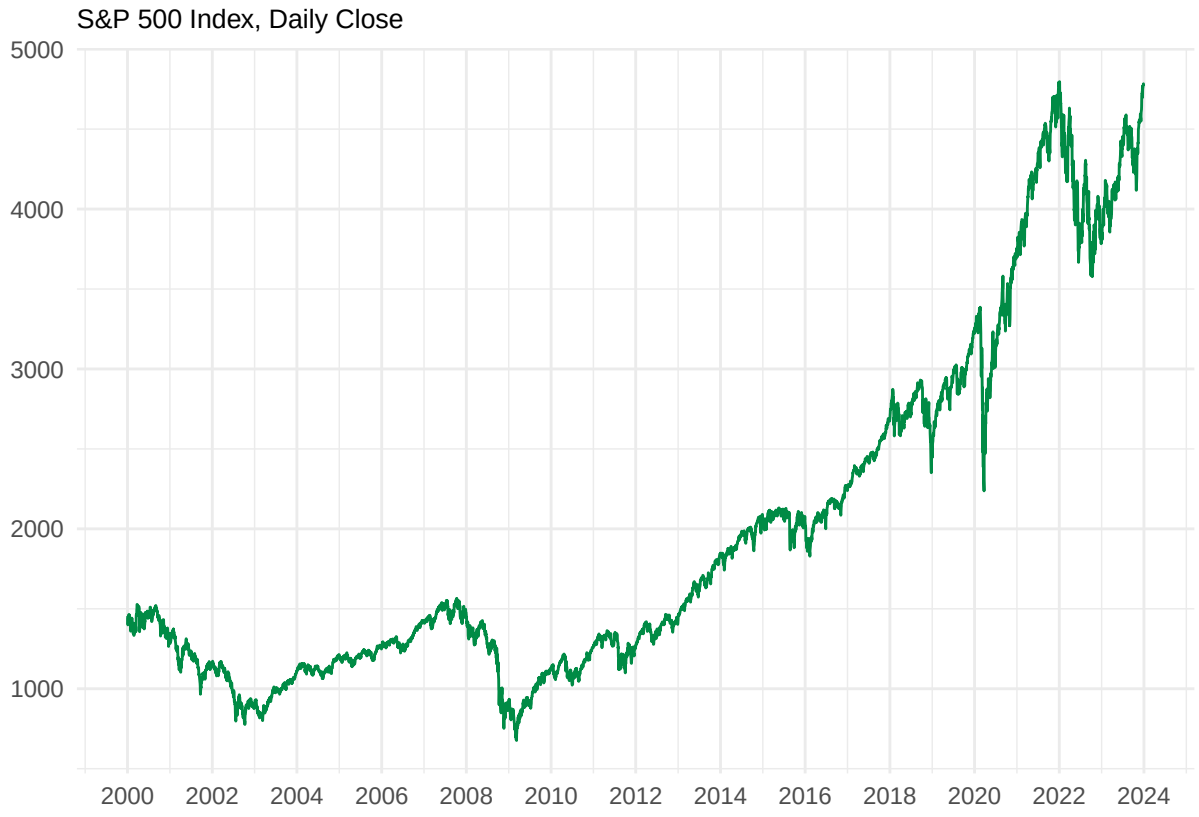
The series is adjusted to logarithmic returns (aka. continuously compounded returns), **in percentage**. This difference measures the percentage change in logarithmic terms, which is often preferred in financial analysis because it treats gains and losses symmetrically.

It is calculated by taking the difference between consecutive log-transformed prices:

$$r_t = log(P_t) - log(P_{t-1}) = log(\frac{P_t}{P_{t-1}})$$

where P_t is the S&P 500 index at time t.

2.2 Data plots and exploratory analysis



Two key takeaways from the EDA are:

- **Volatility clustering**
  - Volatility clustering implies periods of high volatility tend to be followed by high volatility, and periods of low volatility follow low volatility. This characteristic means that large changes tend to cluster together. It is crucial for financial modeling and risk management because it indicates that 'calm' or 'storm' periods tend to persist, which can affect risk assessments.
  - We observed evidence of volatility clustering in the squared returns plot (Fig. 2), for there are several significant spikes (large squared returns). After periods of a spike in volatility, there are typically subsequent periods of large volatility, regardless of the return direction. Similarly, lower areas suggest more stable times with lesser price movement.
  - In financial time series data, the autocorrelation of asset returns' variance is typically much and more persistent than in asset returns mean. This characteristic, as illustrated by *Fig. 3*, is also a sign of volatility clustering.
- **Leptokurtosis (heavy tails)**
  - In financial return data, heavy tails indicate a higher probability of extreme outcomes.
- We observed in *Fig. 4* that bars extend into the tails beyond the red Gaussian curve. This suggests "heavy tails" in the distribution of returns, meaning that extreme returns occur more frequently than expected under a normal distribution.

Figure 1. shows the daily logged returns of the S&P Index over a span of several years. There are visible spikes of high positive and negative returns.

```
SP500_r_df <- data.frame(Date = index(SP500_r), SP = coredata(SP500_r))

# Plot using ggplot
ggplot(SP500_r_df, aes(x = Date, y = SP)) +
  geom_line() +
  labs(title = "SP Index Logged Returns, Daily", x = "Date", y = "SP (in %)",
       caption = "Figure 1.") +
  scale_x_date(date_breaks = "2 years", date_labels = "%Y") +
  theme(plot.caption.position = "plot",
        plot.caption = element_text(hjust = 0)) # hjust=0 moves caption to left
```

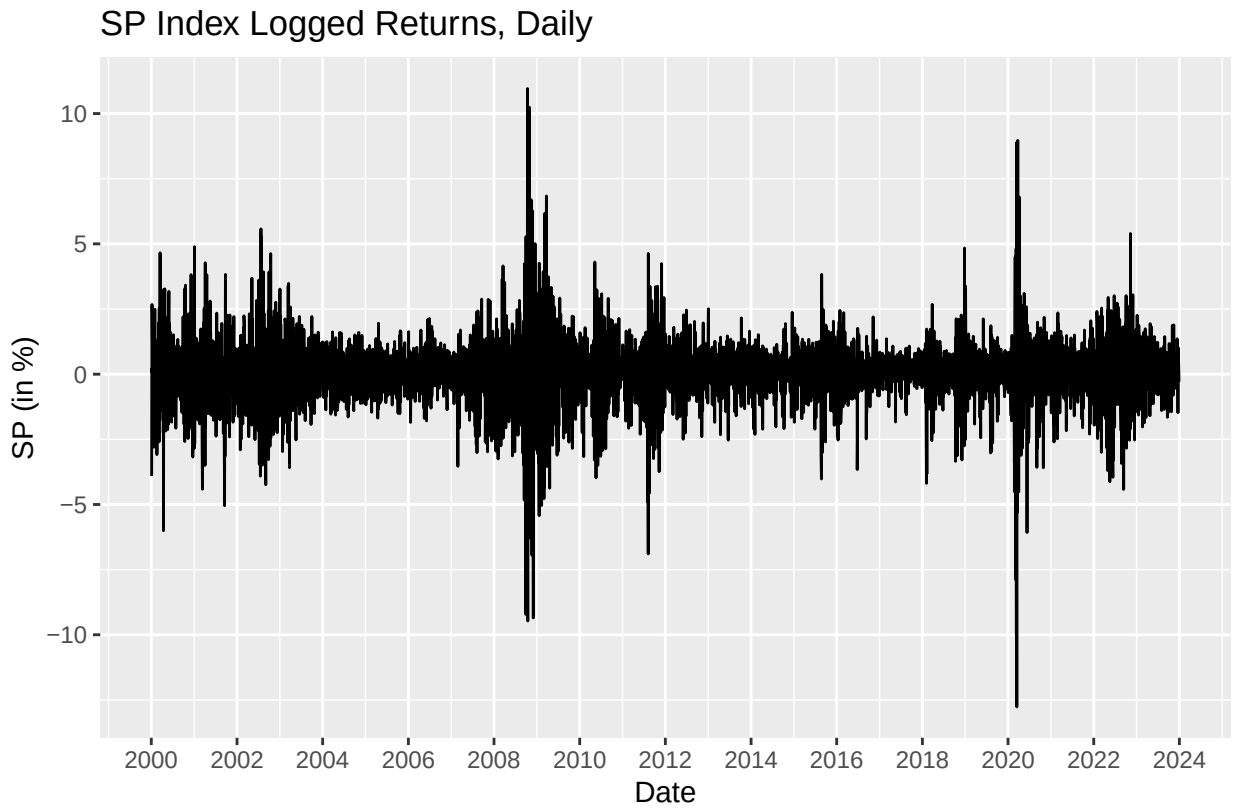


Figure 1.

Figure 2. Shows the approximate sample variance. By squaring the returns, we are able to illustrate the variance of the returns, regardless the sign. This plot displays the magnitude of fluctuations in returns, where peaks indicate high volatility days.

The mean of the returns is very close to zero, therefore we may use the squared return to approximate the sample variance.

```
SP500_r_squared <- SP500_r^2

# Convert the xts object to a data frame for plotting
```

```r
SP500_r_squared_df <- data.frame(Date = index(SP500_r_squared),
                                 SP = coredata(SP500_r_squared))

# Plot using ggplot
ggplot(SP500_r_squared_df, aes(x = Date, y = SP)) +
  geom_line(color = "blue") +
  labs(x = "", y = "", title = "S&P 500 Index Returns Squared, daily Close"
       , caption = "Figure 2.") +
  theme_minimal() +
  theme(plot.title = element_text(size = 10),
        plot.caption.position = "plot",
        plot.caption = element_text(hjust = 0)) +
  scale_x_date(date_breaks = "2 years", date_labels = "%Y")
```

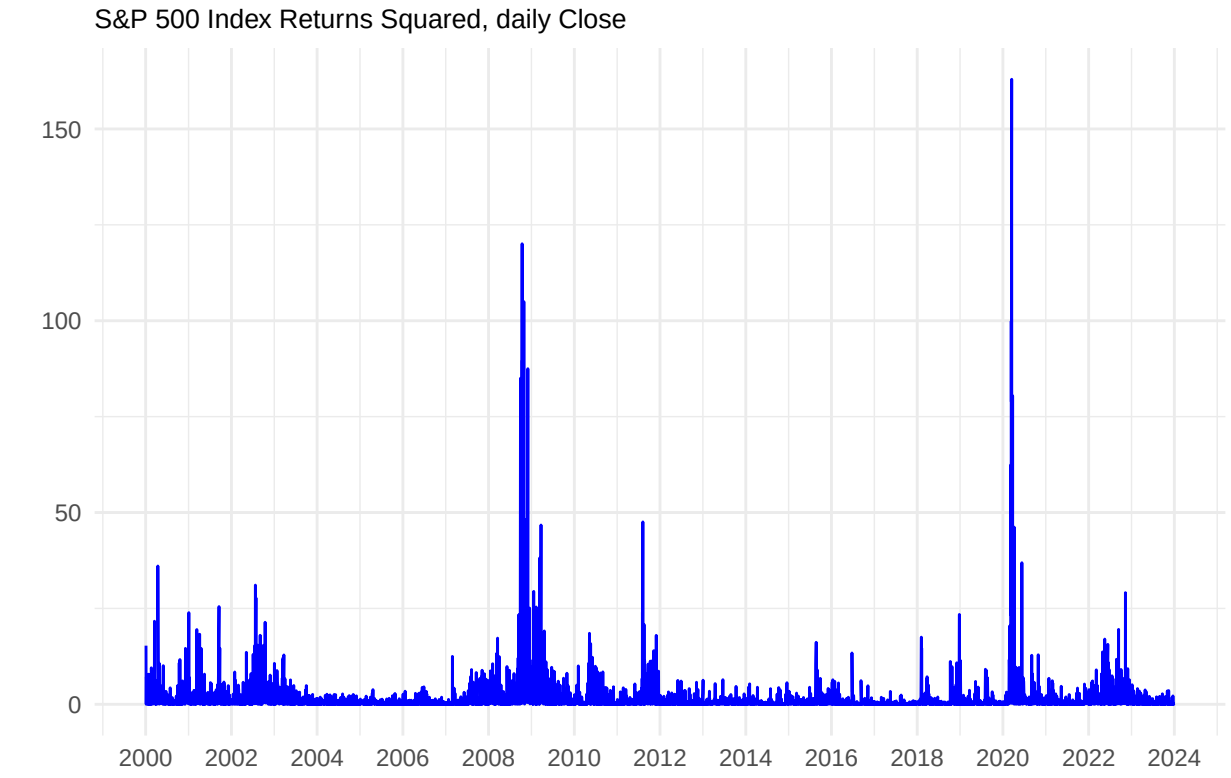


Figure 2.

Figure 3. Is a set of autocorrelation functions (ACF) and partial autocorrelation functions (PACF) plots of the series's sample mean and (approximate) variance.

```r
par(mfrow=c(2,2))  # 2 rows, 2 col
lag_max <- 40
ylu <- c(-0.1,0.4) # y-axis lower and upper bounds

ACF1 <- stats::acf(SP500_r$SP, lag.max=lag_max, plot=T)
PACF1 <- stats::pacf(SP500_r$SP, lag.max=lag_max, plot=T)
```

```
ACF2 <- stats::acf(SP500_r_squared_df$SP, lag.max=lag_max, plot=T)
PACF2 <- stats::pacf(SP500_r_squared_df$SP, lag.max=lag_max, plot=T)
```

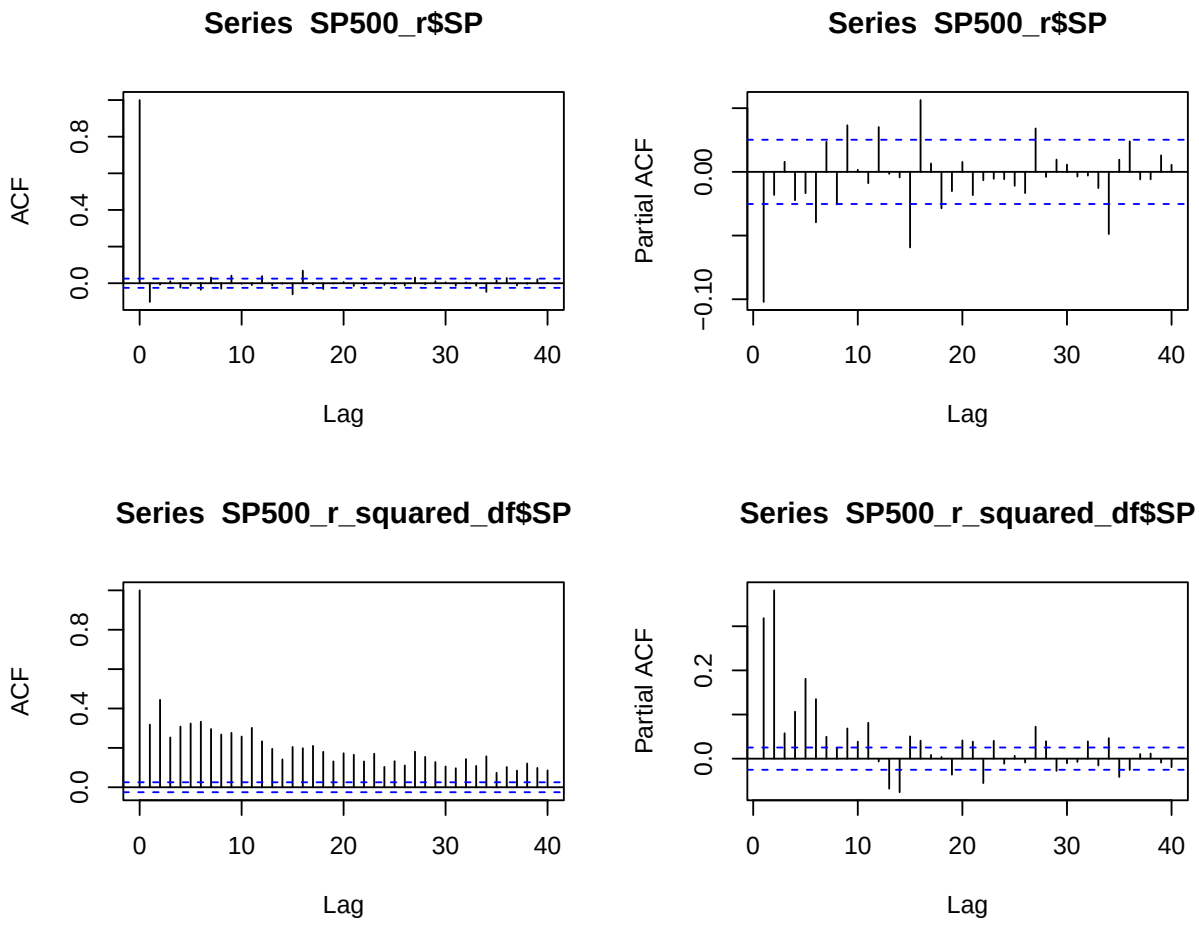
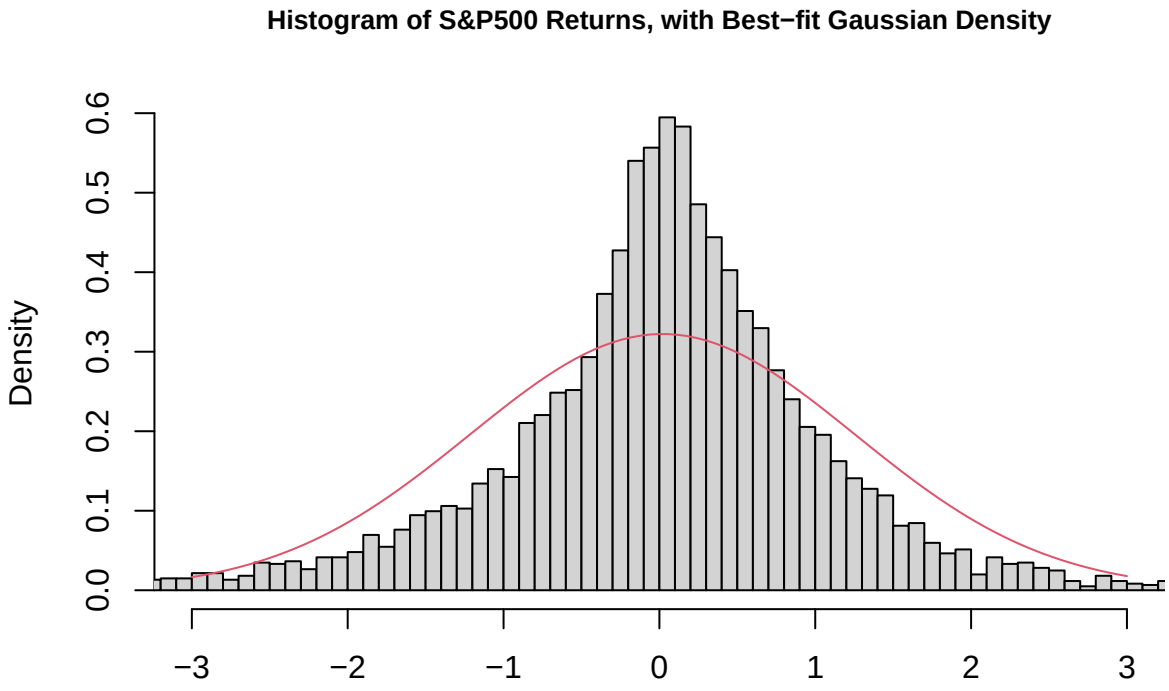


Figure 3. ACF & PACF of sample mean and variance

Figure 4. Shows the histogram on the density of S&P500 returns. The red curve represents the Gaussian (normal) distribution calculated with sample mean and standard deviation. It shows how the data would fit if the distribution is normal.

```
# rm(fit)
vec1 <- as.vector(SP500_r)
fit <- MASS::fitdistr(vec1, "normal")
para <- fit$estimate
graphics::hist(vec1, prob = TRUE, breaks=200, xlim=c(-3,3), col="lightgrey",
cex.main=0.8, main="Histogram of S&P500 Returns, with Best-fit Gaussian Density", xlab='')
graphics::curve(dnorm(x, para[1], para[2]), col = 2, add = TRUE)
```




```
graphics::hist(vec1, prob = TRUE, breaks=500, ylim = c(0, 0.2), xlim=c(-5,-1), col="lightgrey",
cex.main=0.8, main="Detail on the Left Tail (-5% return) ", xlab='')
graphics::curve(dnorm(x, para[1], para[2]), col = 2, add = TRUE)
```

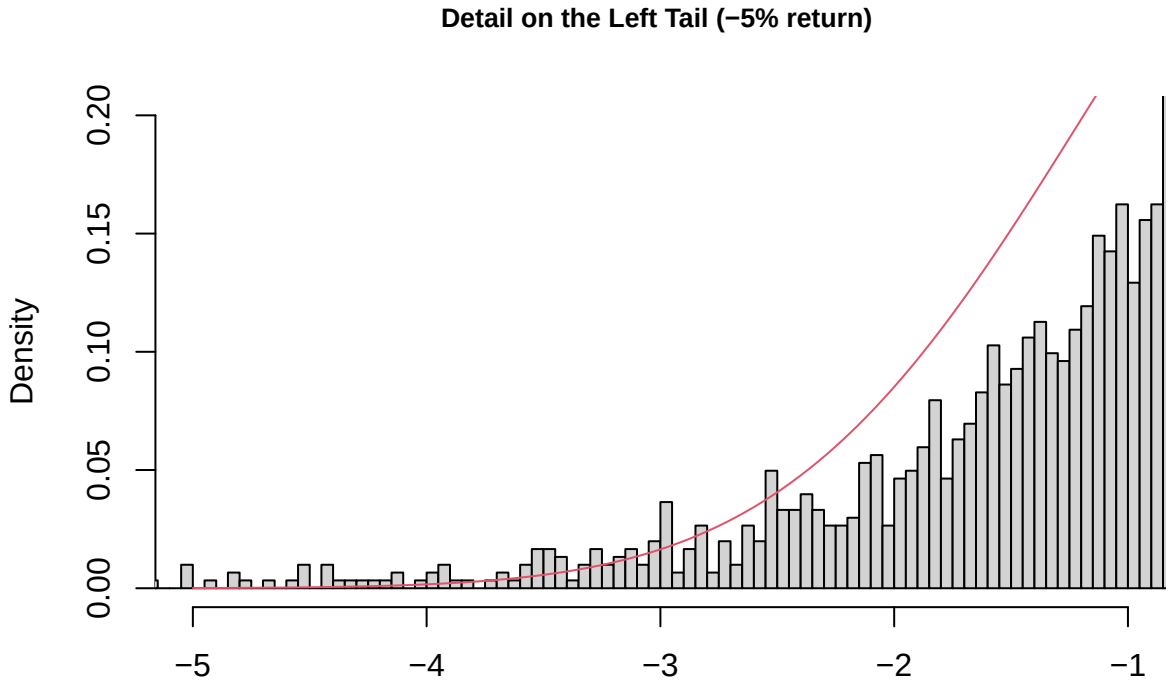


Figure 4. Histogram of S&P500 Returns

2.3. Choice of methods.

An Autoregressive Conditional Heteroskedasticity (ARCH) model is designed for time series data that exhibits non-constant variance and heavy tails.

Heteroskedasticity refers to the presence of time-varying volatility in the series. An ARCH model assumes that volatility depends on past values of the innovations (or residuals). In practice, this often manifests as volatility clustering, which is exactly what we observed in the data plots.

$$ARCH(p) \text{ model of order p: } \sigma^2_{t|t-1} = \alpha_0 + \alpha_1\epsilon^2_{t-1} + \alpha_2\epsilon^2_{t-2} + ... + \alpha_p\epsilon^2_{t-p}$$

where the conditional variance at time t, $\sigma^2_{t|t-1}$ is regressed with the error term ϵ_t, which depends on random innovation z_t: $\epsilon_t = \sigma_{t|t-1}z_t$.

Kurtosis is defined as the scaled 4th momentum of a distribution. It is a measure of the fatness of the tails of a distribution. For a random variable x_t with mean μ, the kurtosis is defined as:

$$\kappa(x_t) = \frac{E(x_t - \mu)^4}{[E(x_t - \mu)^2]^2}$$

The distribution should have $\kappa(x_t) = 3$ if it is normal; any $\kappa(x_t) > 3$ is considered leptokurtic. And an ARCH series, ARCH(1) series for example, assumes leptokurtosis on

$$\kappa(\sigma_{t|t-1}) = 3\frac{1 - \alpha_1^2}{1 - 3a_1^2} > 3$$

However, for the S&P 500 series specifically, we need a Generalized Autoregressive Conditional Heteroskedasticity (GARCH) model.

As illustrated by *Fig. 3*, the ACF of sample variance demonstrated long dependency, where the ACF decays to 0 very slowly, which means we will need a large number of parameters to fit an ARCH model.

On the other hand, a GARCH model can achieve a comparable model with less parameters than ARCH. The GARCH model improves the ARCH model by incorporating not only past innovations but also past conditional variances. This allows is to utilize more information to capture the dynamics of data.

$$GARCH(p, q) \text{ model of order } (p, q):$$
$$\sigma^2_{t|t-1} = \omega + \alpha_1\epsilon^2_{t-1} + \alpha_2\epsilon^2_{t-2} + ... + \alpha_p\epsilon^2_{t-p} + \beta_1\sigma^2_{t-1|t-2} + \beta_2\sigma^2_{t-2|t-3} + ... + +\beta_q\sigma^2_{t-q|t-q-1}$$

We first build a mean model (ARMA model) to account for the predictable structure in the series and extract the residuals. Then, we sequentially build a variance model (GARCH model) based on those residuals to model the volatility.

2.4 Selection of ARMA model.

Initial identification of dependent orders, through the ACF and PACF plots:

- According to *Fig. 3*, the ACF shows a significant spike at lag 1, followed by a rapid decay. This suggests a MA(1) order could be a good fit.
- The PACF also shows a significant spike at lag 1 and then quickly drops off to near-zero values. This is indicative of an of AR(1).
- We could try fitting an ARIMA(1,1) model.

```
m1 <- stats::arima(SP500_r, order=c(1,0,1))
m1
```

```
##
## Call:
## stats::arima(x = SP500_r, order = c(1, 0, 1))
##
## Coefficients:
##          ar1      ma1  intercept
##       0.0555  -0.1597     0.0199
## s.e.  0.1266   0.1252     0.0141
##
## sigma^2 estimated as 1.516:  log likelihood = -9820.13,  aic = 19648.25
```

Model selection using auto.arima()

```
m2 <- forecast::auto.arima(SP500_r)
m2
```

```
## Series: SP500_r
## ARIMA(2,0,1) with zero mean
##
## Coefficients:
##           ar1      ar2     ma1
##       -0.8943  -0.1019  0.7911
## s.e.   0.0773   0.0135  0.0766
##
## sigma^2 = 1.516:  log likelihood = -9818.3
## AIC=19644.6   AICc=19644.6   BIC=19671.42
```

The ARIMA(2,0,1) model has a lower AIC than ARIMA(1,0,1). Therefore we opt to proceed with ARIMA(2,0,1), or equivilently, ARMA(2,1) structure to account for the series's mean.

2.5 GARCH model selection.

```
# Define the range of p and q to iterate over
p_vals <- 1:4
q_vals <- 1:4

# An empty list to store the models' criterion
results <- list()

for (p in p_vals) {
  for (q in q_vals) {

    # specify the GARCH structure
    model <- rugarch::ugarchspec(variance.model = list(model = "sGARCH", garchOrder = c(p, q)),
                                 mean.model = list(armaOrder = c(2, 1)))  # You can modify armaOrder

    # Fit the model
    fit <- tryCatch(rugarch::ugarchfit(spec = model, data = SP500_r), error = function(e) NULL)

    # # If the model fitting was successful, extract the information criterion
    # if (!is.null(fit)) {
      aic <- infocriteria(fit)[1]  # Extract AIC
      bic <- infocriteria(fit)[2]

      # Store the results: (p, q) and AIC in a list
      results[[paste("p", p, "q", q, sep = "_")]] <- list(p = p, q = q, AIC = aic, BIC = bic)
    # }
  }
}
```

```
# Convert the list to a data frame for easier manipulation
results_df <- do.call(rbind, lapply(results, as.data.frame))

results_df
```

```
##         p q      AIC      BIC
## p_1_q_1 1 1 2.766956 2.774733
## p_1_q_2 1 2 2.767287 2.776175
## p_1_q_3 1 3 2.767528 2.777526
## p_1_q_4 1 4 2.767765 2.778874
## p_2_q_1 2 1 2.765381 2.774269
## p_2_q_2 2 2 2.765181 2.775179
## p_2_q_3 2 3 2.765550 2.776659
## p_2_q_4 2 4 2.765690 2.777910
## p_3_q_1 3 1 2.765882 2.775880
## p_3_q_2 3 2 2.765269 2.776378
## p_3_q_3 3 3 2.765549 2.777769
## p_3_q_4 3 4 2.765911 2.779242
## p_4_q_1 4 1 2.766104 2.777213
## p_4_q_2 4 2 2.765592 2.777812
## p_4_q_3 4 3 2.765921 2.779252
## p_4_q_4 4 4 2.766039 2.780481
```

The GARCH(2,1) and GARCH(2,2) models yield the lowest AIC and BIC values. However, the differences in the information criteria are not significant (within 0.001). Following the principle of "simpler is better" and prioritizing model stability, we opt for the more parsimonious GARCH(1,1) model. Further validation and discussion of the model's performance will be addressed in the next section.

3. Reslts and Discussion

The key tests and criteria used in validating a GARCH model are:

1. **Engle's ARCH Test**: To check for ARCH effects in residuals. H0: No ARCH effects in residuals.
2. **Ljung-Box Test**: To test for autocorrelation in standardized and squared residuals. H0: No autocorrelation in residuals.
3. **Weighted ARCH LM Test**: To assess the presence of ARCH effects after fitting.
4. **Nyblom Stability Test**: To detect structural changes in the model's parameters.

```
mfin <- rugarch::ugarchspec(variance.model = list(model="sGARCH", garchOrder=c(1, 1)),
                            mean.model = list(armaOrder=c(2, 1)))

mfin_fit <- rugarch::ugarchfit(spec=mfin, data=SP500_r)
mfin_fit
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : sGARCH(1,1)
## Mean Model   : ARFIMA(2,0,1)
## Distribution : norm
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error    t value Pr(>|t|)
## mu      0.061567    0.007124     8.6426 0.000000
## ar1     0.901935    0.014730    61.2331 0.000000
## ar2     0.039522    0.015311     2.5813 0.009842
## ma1    -0.961454    0.000781 -1231.1236 0.000000
## omega   0.022284    0.002882     7.7315 0.000000
## alpha1  0.120388    0.009270    12.9874 0.000000
## beta1   0.863902    0.009600    89.9880 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error    t value Pr(>|t|)
## mu      0.061567    0.007797     7.8959 0.000000
## ar1     0.901935    0.014424    62.5299 0.000000
## ar2     0.039522    0.015810     2.4997 0.012428
## ma1    -0.961454    0.000470 -2045.3624 0.000000
## omega   0.022284    0.004836     4.6081 0.000004
## alpha1  0.120388    0.015210     7.9151 0.000000
## beta1   0.863902    0.015525    55.6472 0.000000
##
## LogLikelihood : -8343.674
##
## Information Criteria
```

```
## ------------------------------------
##
## Akaike       2.7670
## Bayes        2.7747
## Shibata      2.7670
## Hannan-Quinn 2.7697
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                            statistic p-value
## Lag[1]                         1.116  0.2907
## Lag[2*(p+q)+(p+q)-1][8]        3.157  0.9921
## Lag[4*(p+q)+(p+q)-1][14]       6.515  0.6555
## d.o.f=3
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                           statistic p-value
## Lag[1]                       0.9256  0.3360
## Lag[2*(p+q)+(p+q)-1][5]      5.2484  0.1344
## Lag[4*(p+q)+(p+q)-1][9]      6.7999  0.2165
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale P-Value
## ARCH Lag[3]    0.1441 0.500 2.000  0.7042
## ARCH Lag[5]    1.8296 1.440 1.667  0.5101
## ARCH Lag[7]    2.4696 2.315 1.543  0.6187
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  3.316
## Individual Statistics:
## mu     0.59429
## ar1    0.07962
## ar2    0.08025
## ma1    0.14684
## omega  0.13018
## alpha1 0.25460
## beta1  0.60522
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          1.69 1.9 2.35
## Individual Statistic:     0.35 0.47 0.75
##
## Sign Bias Test
## ------------------------------------
##                    t-value      prob sig
## Sign Bias            3.684 2.315e-04 ***
## Negative Sign Bias   0.903 3.666e-01
## Positive Sign Bias   2.576 1.003e-02  **
## Joint Effect        43.864 1.613e-09 ***
```

```
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     210.9    2.253e-34
## 2    30     227.7    1.012e-32
## 3    40     270.9    1.726e-36
## 4    50     306.3    6.504e-39
##
##
## Elapsed time : 0.376713
```

The results of the GARCH(1,1) model indicate strong statistical significance for all estimated parameters, with very low p-values (< 0.0001).

- **ARCH-LM Test**: The null hypothesis that there are no remaining ARCH effects is not rejected (p-values > 0.5), suggesting that the model effectively accounts for heteroskedasticity.
- **Ljung-Box Test**: The null hypothesis that no autocorrelation remains in the residuals is not rejected, as the p-values are large (0.29, 0.99), meaning the model has adequately captured the time series' dynamics.
- **Nyblom Stability Test**: The joint statistic exceeds the 5% critical value, indicating possible parameter instability, though individual statistics mostly fall within acceptable ranges.

To summarize, the model demonstrate some instability over time, as well as some flaws in fitting the data. But the model estimates are significant; the model's residuals satisfy whiten noise assumption; it adequately captured the volatility dynamics of the series effectively.

4. Forecasts: Value at Risk and Expected Shortfall

Forecast the Value at Risk (VaR) at alpha=1% for a $10,000 position in the Index for the next 8 days.

The ugarchforecast() fucntion returns the forecasted mean and variance with specified forecast horizon.

```
n_future <- 8
model_forecast <- rugarch::ugarchforecast(fit=mfin_fit, n.ahead=n_future)
model_forecast #%>% class()
```

```
##
## *----------------------------------*
## *       GARCH Model Forecast       *
## *----------------------------------*
## Model: sGARCH
## Horizon: 8
## Roll Steps: 0
## Out of Sample: 0
##
## 0-roll forecast [T0=2023-12-29]:
##      Series  Sigma
## T+1 0.02139 0.6768
## T+2 0.01171 0.6879
## T+3 0.01501 0.6986
## T+4 0.01760 0.7090
## T+5 0.02007 0.7190
## T+6 0.02241 0.7288
## T+7 0.02461 0.7383
## T+8 0.02668 0.7476
```

One of the application is to predict value at risk (VaR) and Expected Shortfall (ES). The VaR of a series at the $\alpha\%$ confidence level describes the maximum loss incurred in a predefined period of time and confidence level α. It represents the maximum potential loss at a certain confidence level, with a probability of $1\check{}\alpha$.

It is defined by formula $r_t^{VaR(\alpha\%)} = \mu_{t|t-1} - z_{\alpha\%} * \sigma_{t|t-1}$,
or in statistical terms, the α quantile.

Where μ and $\sigma_{t|t-1}$ can be captured by the model, and $z_{\alpha\%}$ by the standard normal.

Let's say we were to estimate VaR at $\alpha = 1$ for a $10,000 position in S&P500.

```
mean_f <- as.numeric(model_forecast@forecast$seriesFor)/100 # from % to fraction
std_f <- as.numeric(model_forecast@forecast$sigmaFor)/100 # from % to fraction]

zv <-qnorm(0.99, mean=0,sd=1)

# Our variable of interest is an expected return, lets call it rt.
rt_VaR_1percent <- abs(mean_f - zv*std_f)

dollar_VaR_1percent = rt_VaR_1percent*10000

# dollar_VaR_1percent %>% class # This is alsp a xts/zoo, same as the series of data

cat("Return value at risk: ",rt_VaR_1percent)
```

```
## Return value at risk:  0.01553131 0.01588529 0.01610145 0.01631708 0.01652675 0.01673097 0.01692998 0
```

```
cat("\nDollar value at risk: ", dollar_VaR_1percent)
```

```
##
## Dollar value at risk:  155.3131 158.8529 161.0145 163.1708 165.2675 167.3097 169.2998 171.2401
```

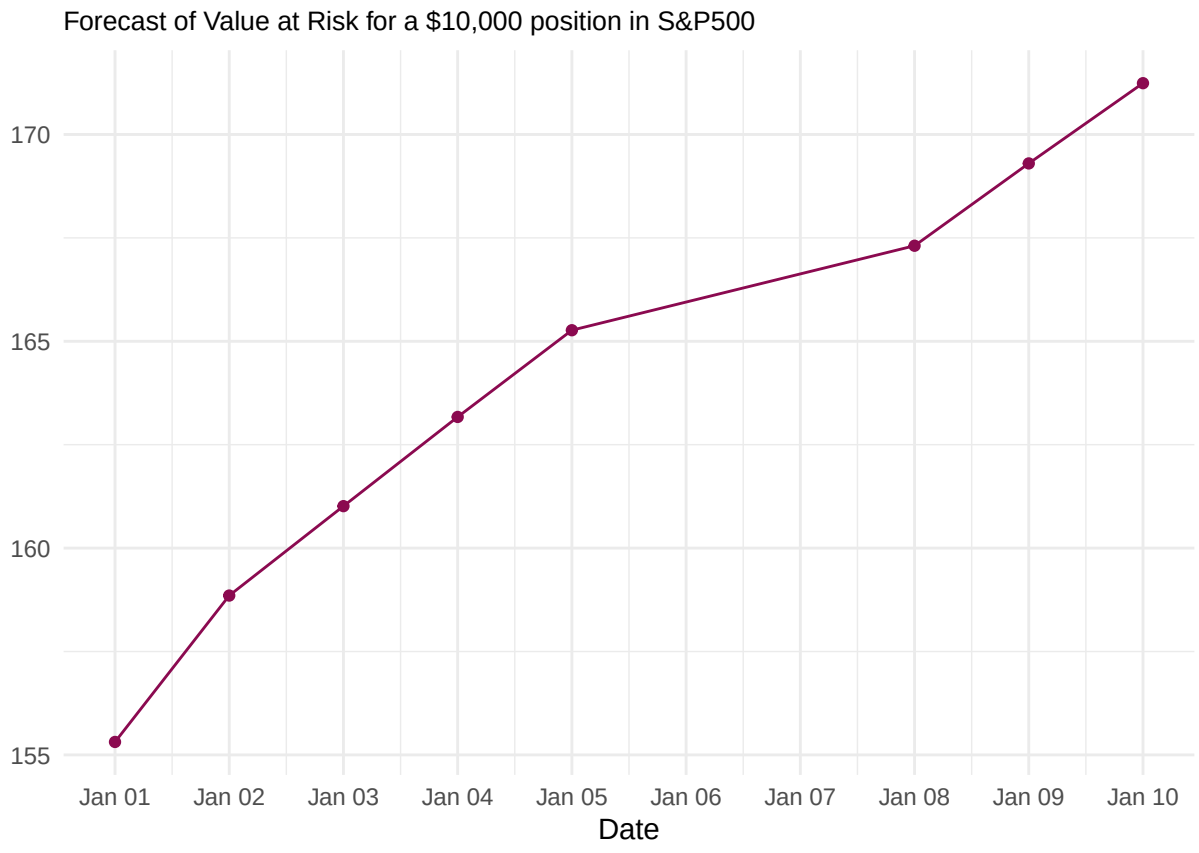
- **dollar_VaR_1percent**: represents the dollar value at risk for the given position size of $50,000.

Plot the forecasts:

```
id <- timetk::tk_index(SP500_r)
```

```
## Warning in .check_tzones(e1, e2): 'tzone' attributes are inconsistent
```

```
id_f <- timetk::tk_make_future_timeseries(id, length_out=n_future, inspect_weekdays=TRUE)
dollar_VaR_1percent <- xts(dollar_VaR_1percent, order.by=id_f)
colnames(dollar_VaR_1percent) <- "VaR"
ggplot(data=dollar_VaR_1percent, aes(x=index(dollar_VaR_1percent), y=VaR)) +
geom_line(color="deeppink4") +
geom_point(color="deeppink4") +
labs(x="Date", y="", title="Forecast of Value at Risk for a $10,000 position in S&P500") +
theme_minimal() + scale_x_date(date_breaks="1 day", date_labels = "%b %d") +
theme(plot.title = element_text(size=10))
```



Forecasting Expected Shortfall

Forecast the ES at alpha=1% for a $10,000 position in the Index for the next 8 days.

While VaR measures the maximum potential lost at a confidence level, the ES captures the average of such loss.

```
zv <-qnorm(0.99, mean=0,sd=1)
es_right = (1/sqrt(2*pi))*exp(-zv^2/2)/0.01

rt_ES_1percent_f <- abs(mean_f - es_right * std_f)
dollar_ES_1percent_f <- rt_ES_1percent_f*10000

dollar_ES_1percent_f <- xts(dollar_ES_1percent_f, order.by=id_f)
colnames(dollar_ES_1percent_f) <- "ES"

cat("Return expected shortfall: ",rt_ES_1percent_f)
```

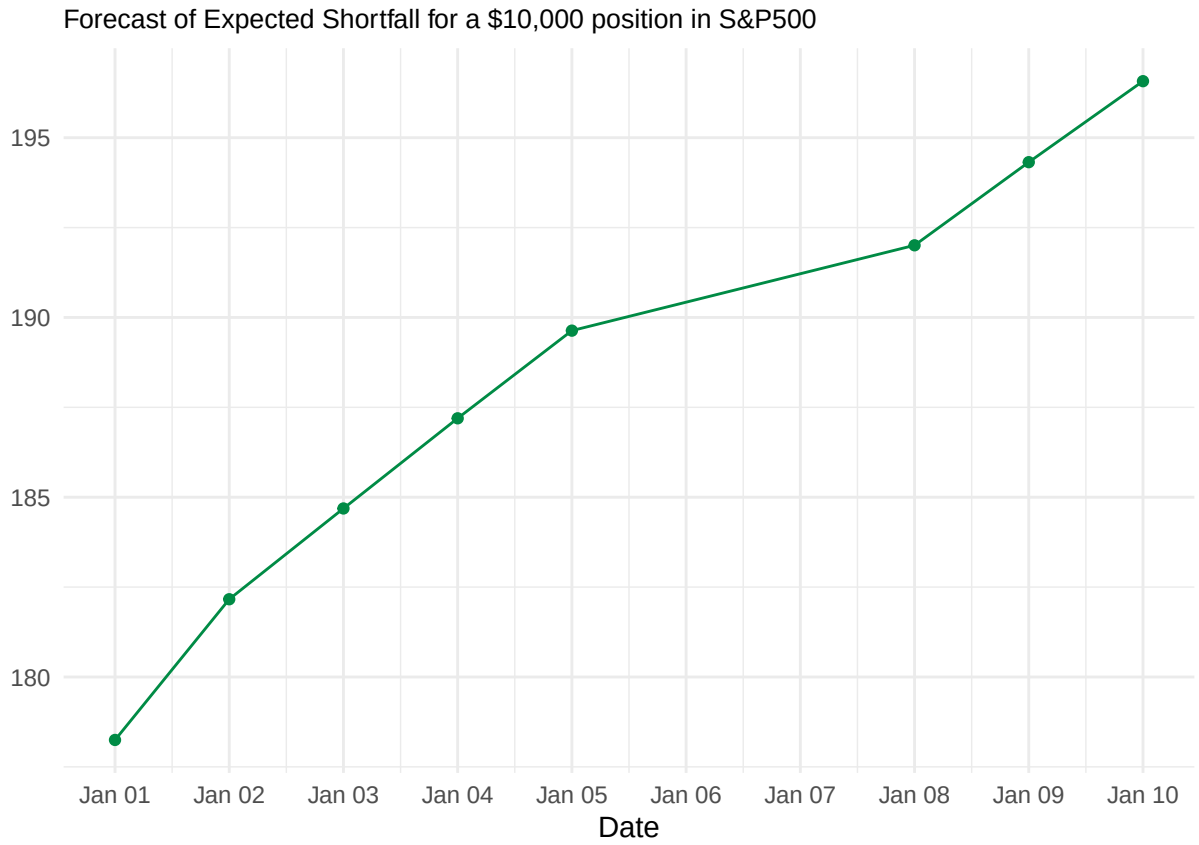
```
## Return expected shortfall:  0.01782482 0.01821626 0.01846872 0.01871954 0.01896335 0.01920071 0.0194
```

```
cat("\nDollar expected shortfall: ", dollar_ES_1percent_f)
```

```
##
## Dollar expected shortfall:  178.2482 182.1626 184.6872 187.1954 189.6335 192.0071 194.3192 196.5724
```

Plot the results:

```
ggplot(data=dollar_ES_1percent_f, aes(x=index(dollar_ES_1percent_f), y=ES)) +
geom_line(color="springgreen4") +
geom_point(color="springgreen4") +
labs(x="Date", y="", title="Forecast of Expected Shortfall for a $10,000 position in S&P500") +
theme_minimal() + scale_x_date(date_breaks="1 day", date_labels = "%b %d") +
theme(plot.title = element_text(size=10))
```


Forecast of Expected Shortfall for a $10,000 position in S&P500
195
190
185
180
Jan 01
Jan 02
Jan 03
Jan 04
Jan 05
Jan 06
Jan 07
Jan 08
Jan 09
Jan 10
Date