Testing

Contents

Data

Raw Data

```r
#political shocks
#raw_truths <- read.csv(here("data/political_data", "trump_all_truths.csv"))
#raw_tweets <- read.csv(here("data/political_data", "tweets.csv"))


#market prices
raw_ONEQ <- read.csv(here("data/market_data", "ONEQ.csv"))
raw_SMI <- read.csv(here("data/market_data", "SMI.csv"))
raw_SPY <- read.csv(here("data/market_data", "SPY.csv"))
raw_SPY2425 <- read.csv(here("data/market_data", "SPY_24_25.csv"))
raw_SPY2021_01 <- read.csv(here("data/market_data", "SPY-2021-01.csv"))
raw_VTHR <- read.csv(here("data/market_data", "VTHR.csv"))
raw_VTI <- read.csv(here("data/market_data", "VTI.csv"))
raw_VGK <- read.csv(here("data/market_data", "VGK.csv"))
raw_DAX <- read.csv(here("data/market_data", "DAX.csv"))
raw_ASHR <- read.csv(here("data/market_data", "ASHR.csv"))

raw_SPYy <- read.csv(here("data/market_data", "Spyqyahoo.csv")) #yahoo
```

Cleaning The Data

```r
#political shocks
truths <- 1
tweets <- 1


#market prices #only cleaning dates for the time being
raw_ONEQ$timestamp = as.POSIXct(raw_ONEQ$timestamp, format = "%Y-%m-%d %H:%M:%S", tz = "EST")
raw_SMI$timestamp = as.POSIXct(raw_SMI$timestamp, format = "%Y-%m-%d %H:%M:%S", tz = "EST")
raw_SPY$timestamp = as.POSIXct(raw_SPY$timestamp, format = "%Y-%m-%d %H:%M:%S", tz = "EST")
raw_VTHR$timestamp = as.POSIXct(raw_VTHR$timestamp, format = "%Y-%m-%d %H:%M:%S", tz = "EST")
raw_VTI$timestamp = as.POSIXct(raw_VTI$timestamp, format = "%Y-%m-%d %H:%M:%S", tz = "EST")
raw_VGK$timestamp = as.POSIXct(raw_VGK$timestamp, format = "%Y-%m-%d %H:%M:%S", tz = "UCT")
raw_DAX$timestamp = as.POSIXct(raw_DAX$timestamp, format = "%Y-%m-%d %H:%M:%S", tz = "UCT")
raw_ASHR$timestamp = as.POSIXct(raw_ASHR$timestamp, format = "%Y-%m-%d %H:%M:%S", tz = "UCT") #fix time
```

Daily Data

```r
#political shocks


#market prices
day_SPY_0409 = filter(raw_SPY, str_detect(timestamp, "^2025-04-09")) #9th of april
day_SPY_0409$timestamp = as.POSIXct(day_SPY_0409$timestamp,
```

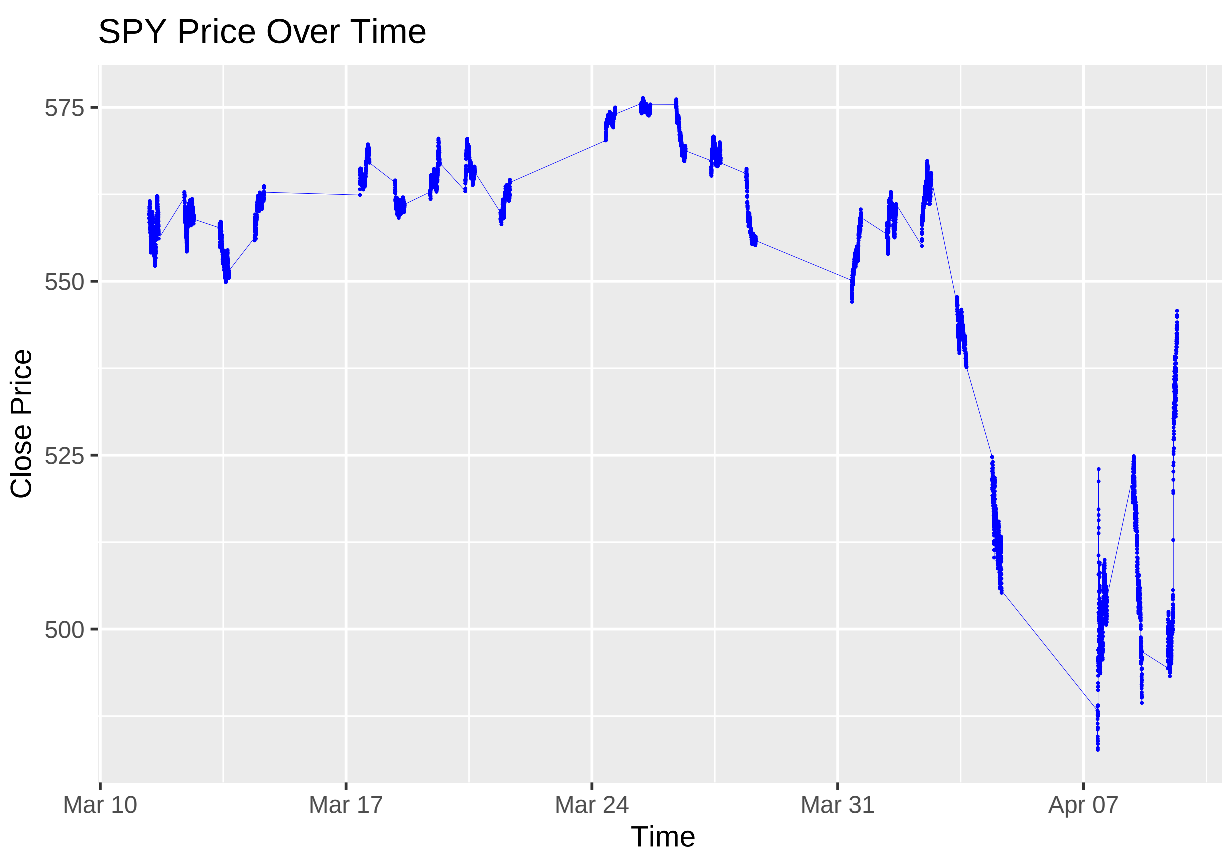
```
                                    format = "%Y-%m-%d %H:%M:%S", tz = "EST")

yahoo_ds0409 = filter(raw_SPYy, str_detect(Date, "^2025-04-09"))
yahoo_ds0409$Date = as.POSIXct(yahoo_ds0409$Date,
                                    format = "%Y-%m-%dT%H:%M:%S", tz = "UTC")
yahoo_ds0409$Date = with_tz(yahoo_ds0409$Date, "EST")
```

Plots

Total

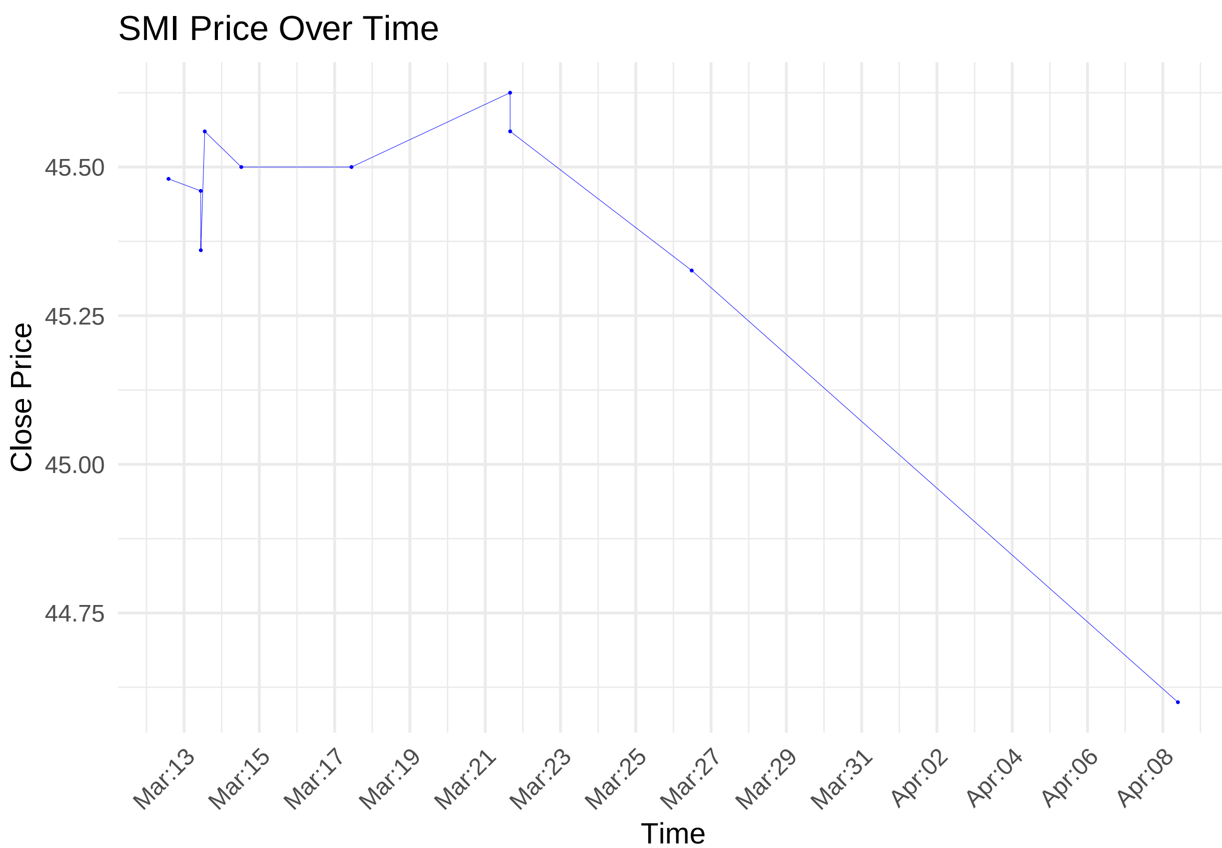
```
#SPY
ggplot(raw_SPY, aes(x = timestamp, y = close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1), color="blue", linewidth=0.05) +
  labs(title = "SPY Price Over Time",
       x = "Time",
       y = "Close Price")
```



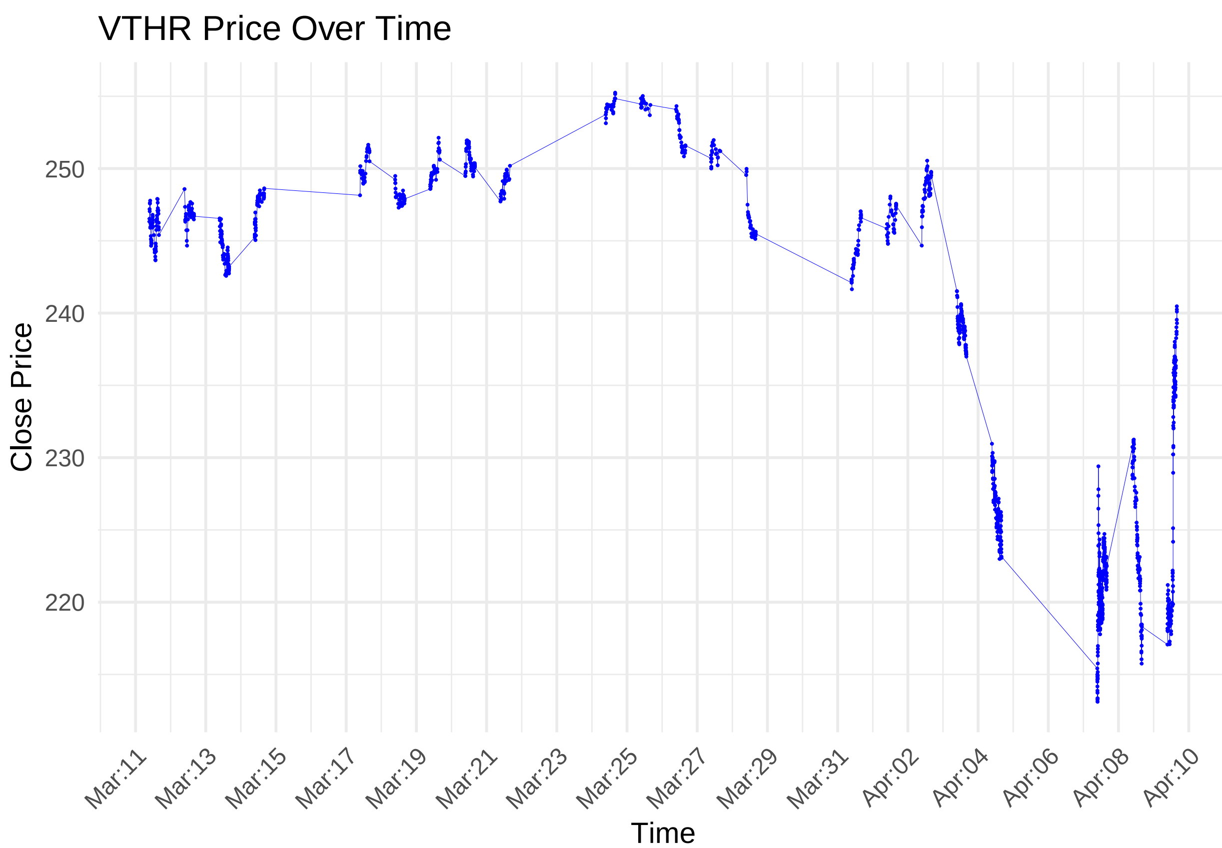

```
#ONEQ
ggplot(raw_ONEQ, aes(x = timestamp, y = close)) +
  geom_point(color = "blue", size = 0.01) +
```

```r
  geom_line(aes(group=1), color="blue", linewidth=0.05) +
  labs(title = "ONEQ Price Over Time",
       x = "Time",
       y = "Close Price") +
   scale_x_datetime(date_labels = "%b:%d",
                    date_breaks = "2 day") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```


```r
#SMI
ggplot(raw_SMI, aes(x = timestamp, y = close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1), color="blue", linewidth=0.05) +
  labs(title = "SMI Price Over Time",
       x = "Time",
       y = "Close Price") +
   scale_x_datetime(date_labels = "%b:%d",
                    date_breaks = "2 day") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

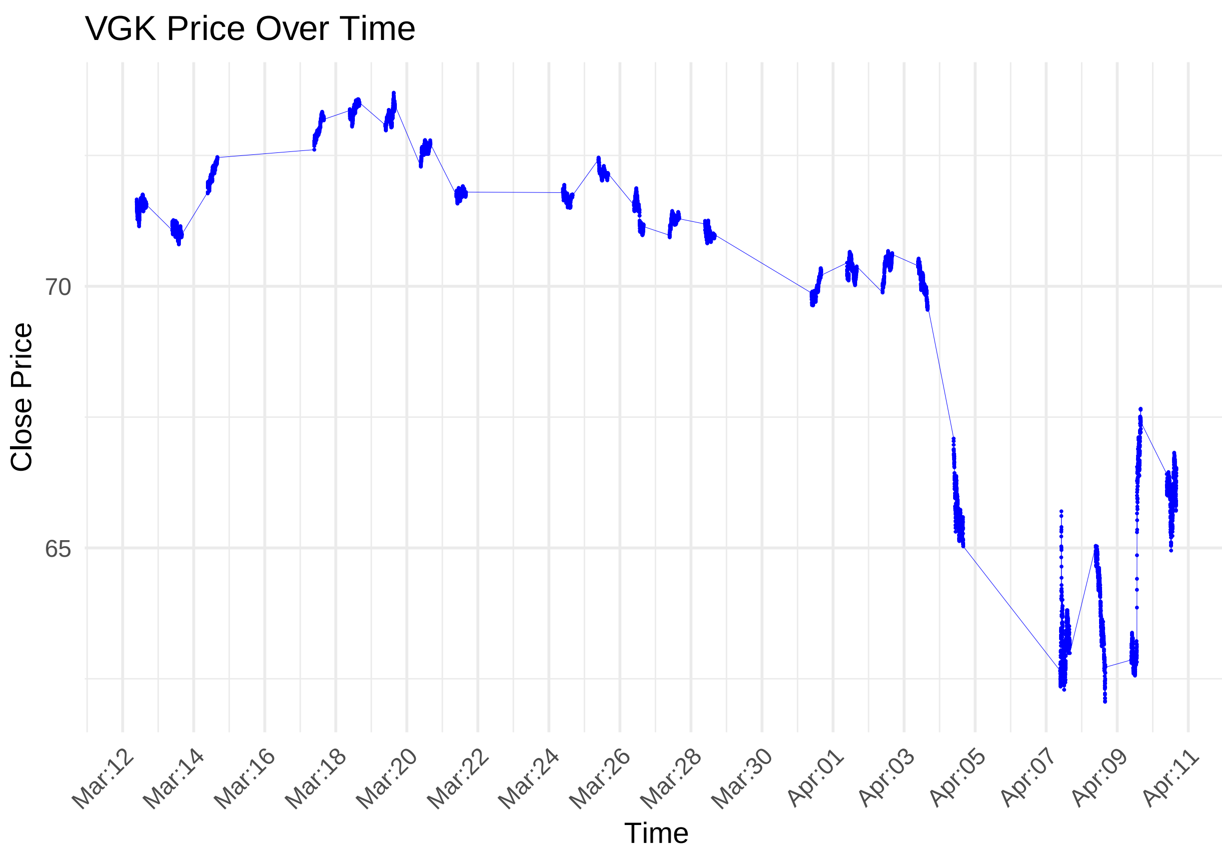



```
#VTHR
ggplot(raw_VTHR, aes(x = timestamp, y = close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1), color="blue", linewidth=0.05) +
  labs(title = "VTHR Price Over Time",
       x = "Time",
       y = "Close Price") +
   scale_x_datetime(date_labels = "%b:%d",
                    date_breaks = "2 day") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

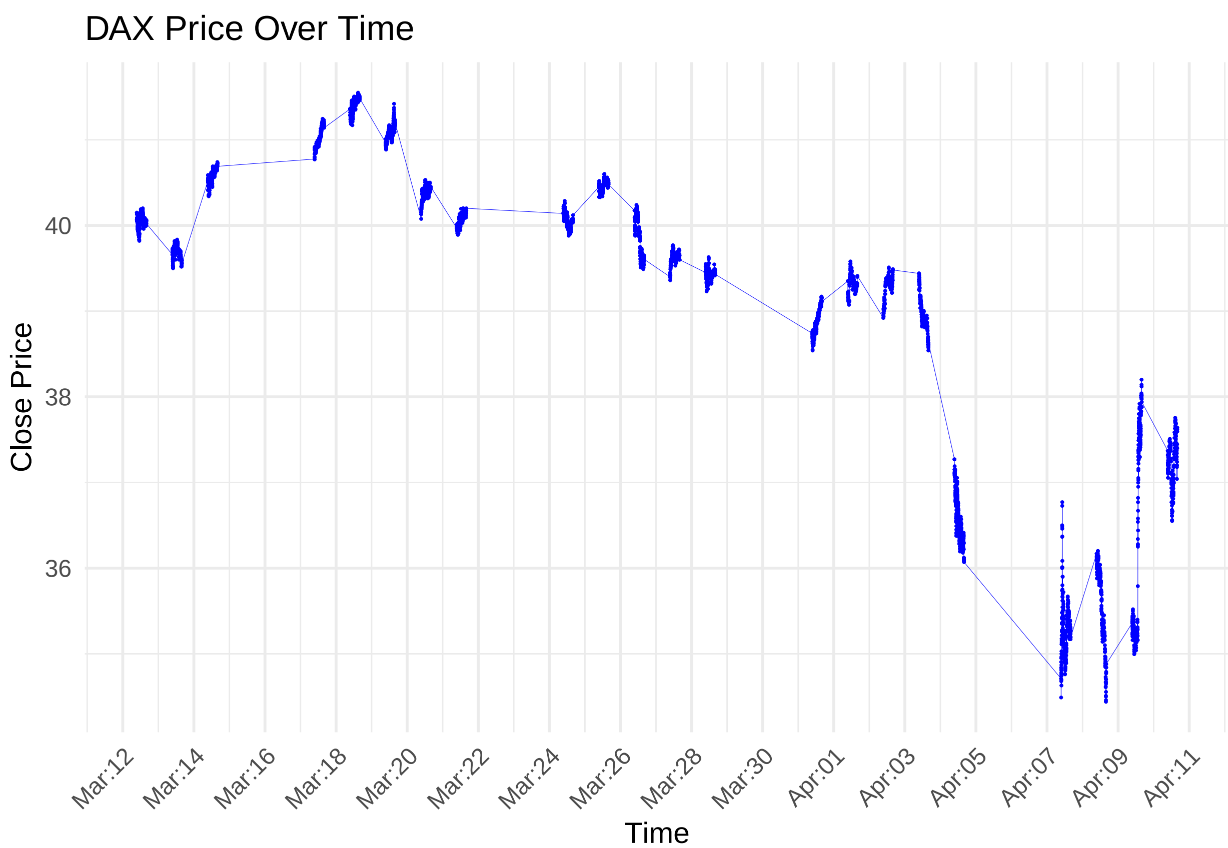



```r
#VTI
ggplot(raw_VTI, aes(x = timestamp, y = close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1), color="blue", linewidth=0.05) +
  labs(title = "VTI Price Over Time",
       x = "Time",
       y = "Close Price") +
   scale_x_datetime(date_labels = "%b:%d",
                    date_breaks = "2 day") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

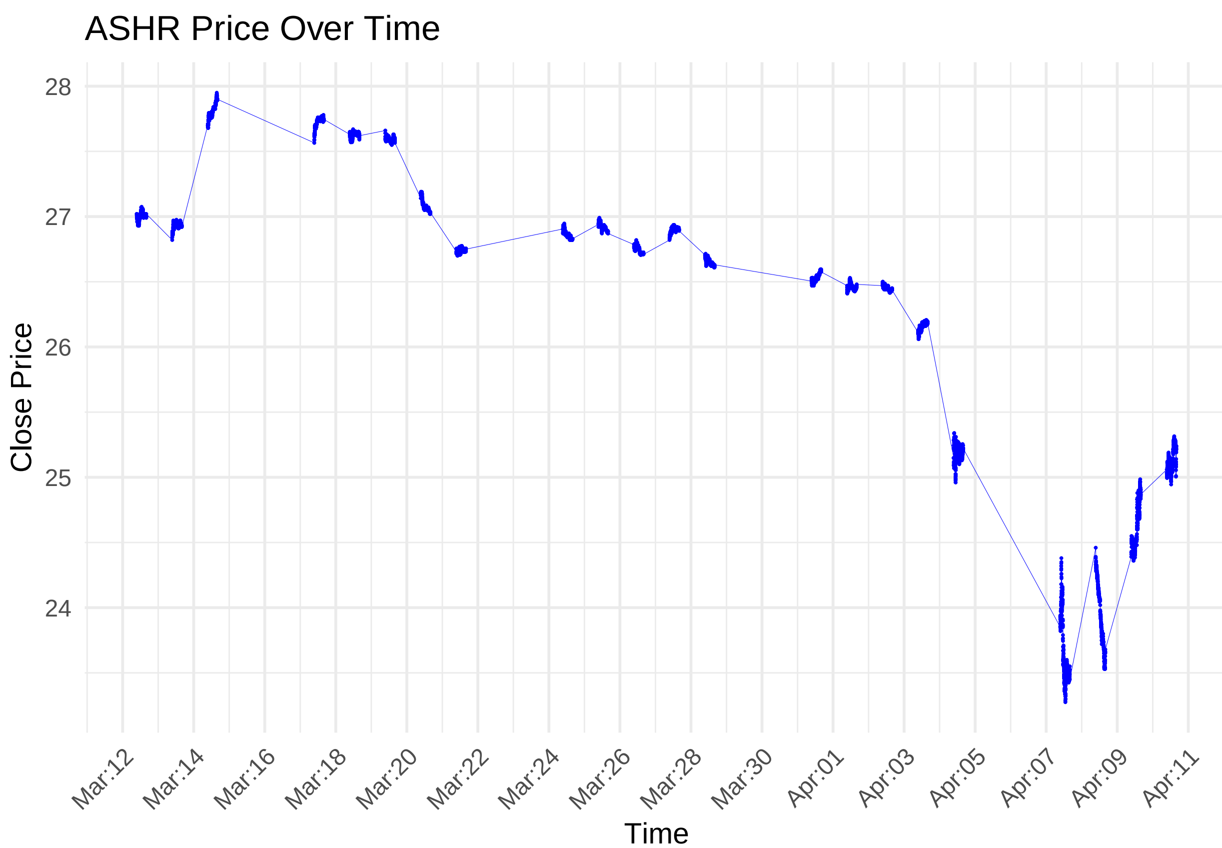

```r
#VGK
ggplot(raw_VGK, aes(x = timestamp, y = close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1), color="blue", linewidth=0.05) +
  labs(title = "VGK Price Over Time",
       x = "Time",
       y = "Close Price") +
   scale_x_datetime(date_labels = "%b:%d",
                    date_breaks = "2 day") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
#DAX
ggplot(raw_DAX, aes(x = timestamp, y = close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1), color="blue", linewidth=0.05) +
  labs(title = "DAX Price Over Time",
       x = "Time",
       y = "Close Price") +
   scale_x_datetime(date_labels = "%b:%d",
                    date_breaks = "2 day") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
#ASHR
ggplot(raw_ASHR, aes(x = timestamp, y = close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1), color="blue", linewidth=0.05) +
  labs(title = "ASHR Price Over Time",
       x = "Time",
       y = "Close Price") +
   scale_x_datetime(date_labels = "%b:%d",
                    date_breaks = "2 day") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



#Get Truths April 9th

Per Day

```
#SPY Source: alpha
ggplot(day_SPY_0409, aes(x = timestamp, y = close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1)) +
  labs(title = "SPY alpha Price April 9th",
       x = "Time",
       y = "Close Price") +
   scale_x_datetime(date_labels = "%H:%M",
                    date_breaks = "60 min") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

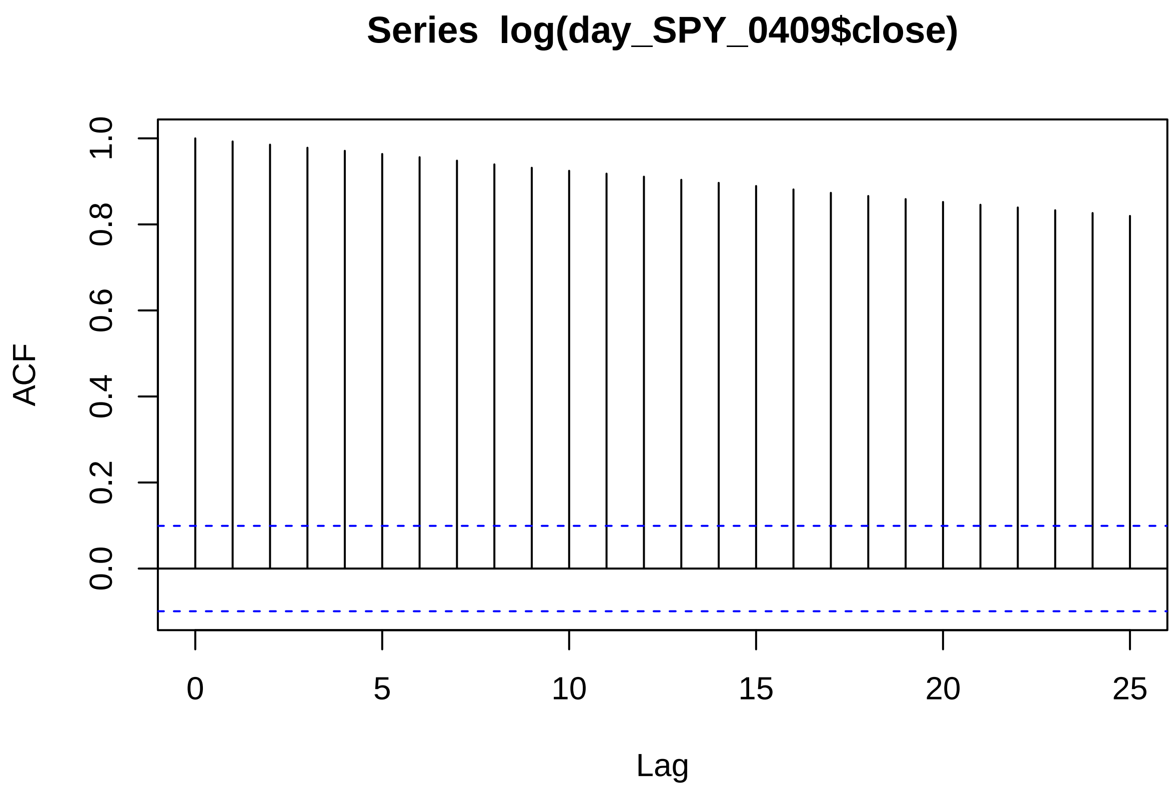

```r
#SPY log prices Source: alpha
ggplot(day_SPY_0409, aes(x = timestamp, y = log(close))) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1)) +
  labs(title = "SPY Log Price April 9th",
       x = "Time",
       y = "Log Close Price") +
   scale_x_datetime(date_labels = "%H:%M",
                    date_breaks = "60 min") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```


```r
#SPY Source: yahoo
ggplot(yahoo_ds0409, aes(x = Date, y = Close)) +
  geom_point(color = "blue", size = 0.01) +
  geom_line(aes(group=1)) +
  labs(title = "SPY yahoo Price April 9th",
       x = "Time",
       y = "Close Price") +
   scale_x_datetime(date_labels = "%H:%M",
                    date_breaks = "60 min") +
   theme_minimal() +
   theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

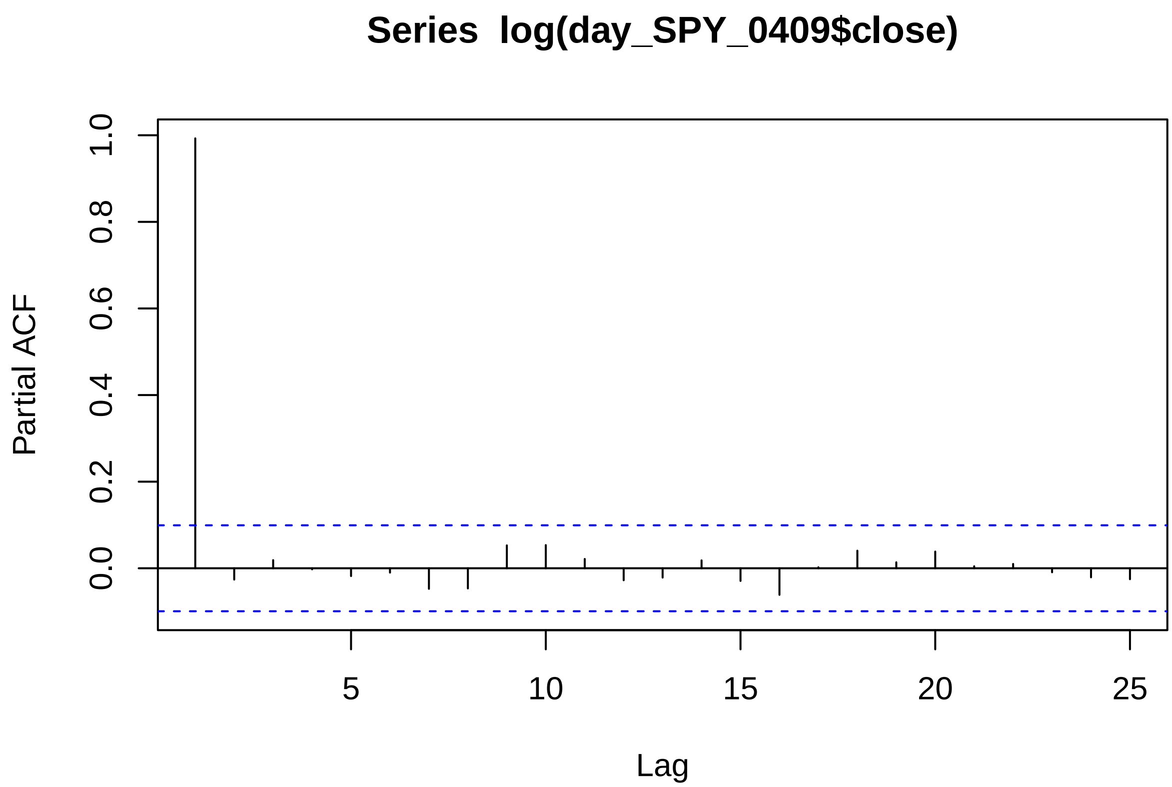
```
## Warning: Removed 3 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

Time Series Analysis

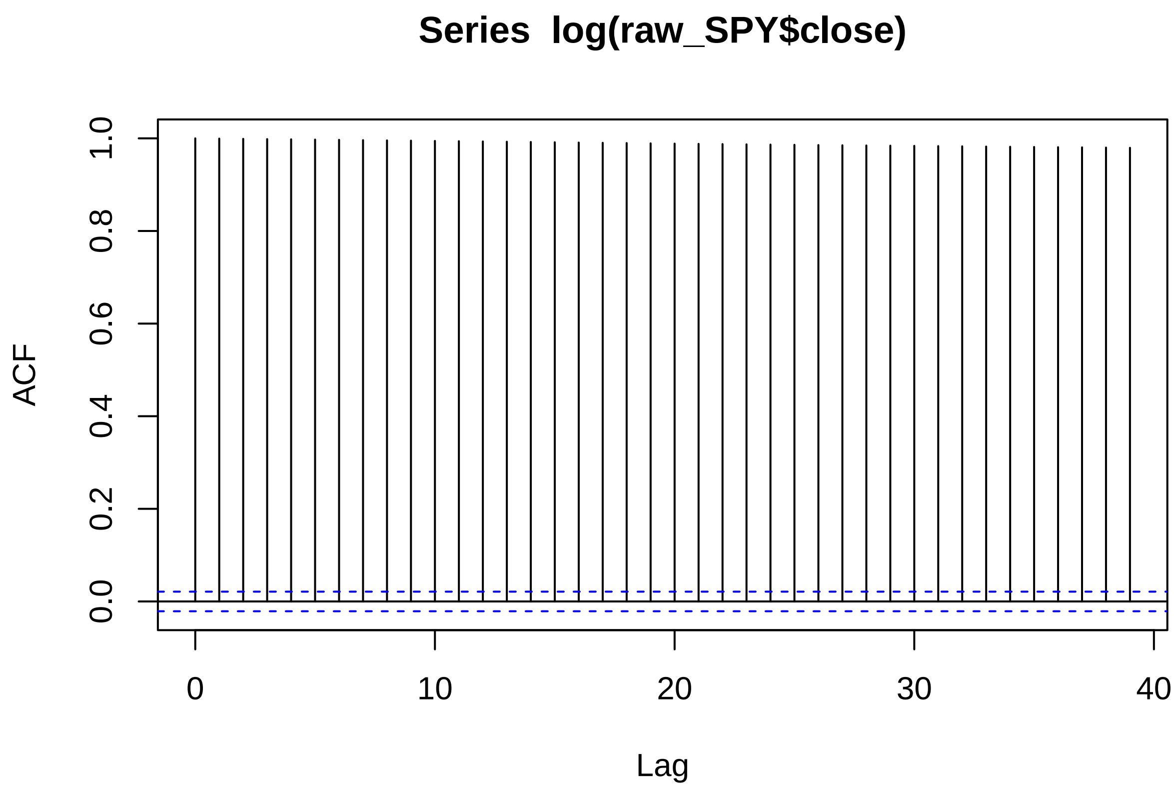
```
acf(log(day_SPY_0409$close))
```



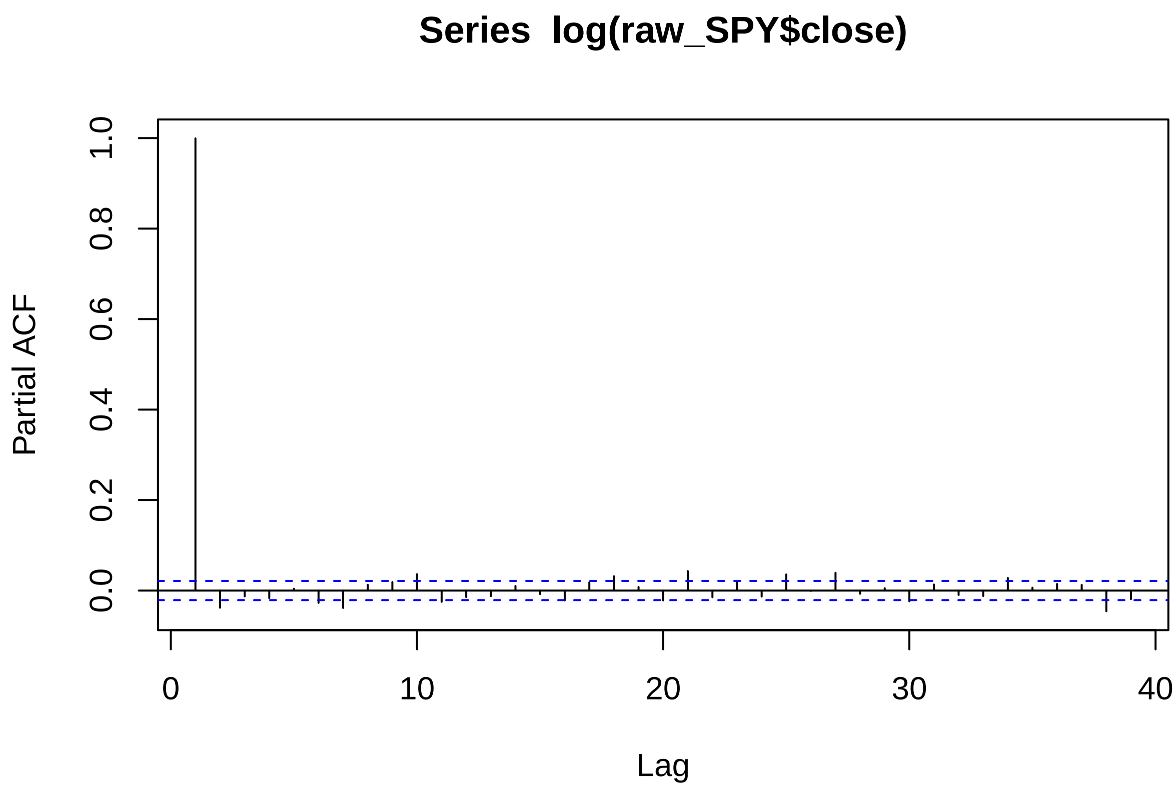

```
pacf(log(day_SPY_0409$close))
```




```
acf(log(raw_SPY$close))
```




```r
pacf(log(raw_SPY$close))
```




```
AR1 = arima(day_SPY_0409$close,c(1,0,0),method="ML")
AR2 = arima(day_SPY_0409$close,c(2,0,0),method="ML")
AR3 = arima(day_SPY_0409$close,c(3,0,0),method="ML")
table1 = export_summs(AR1,AR2,AR3, model.names = c("AR1","AR2","AR3"), digits = 4)
```

```
## Warning in FUN(X[[i]], ...): tidy() does not return p values for models of
## class data.frame; significance stars not printed.
## Warning in FUN(X[[i]], ...): tidy() does not return p values for models of
## class data.frame; significance stars not printed.
## Warning in FUN(X[[i]], ...): tidy() does not return p values for models of
## class data.frame; significance stars not printed.
```

```
huxtable::caption(table1) <- "AR Estimations"
huxtable::set_width(table1, 0.8)
```

```
AR1res = as.numeric(AR1$residuals)
AR1res_lagged <- lag(AR1res, 1)
iidcheck1 = lm(AR1res ~ AR1res_lagged)
AR2res = as.numeric(AR2$residuals)
AR2res_lagged <- lag(AR2res, 1)
iidcheck2 = lm(AR2res ~ AR2res_lagged)
AR3res = as.numeric(AR3$residuals)
AR3res_lagged <- lag(AR3res, 1)
iidcheck3 = lm(AR3res ~ AR3res_lagged)
```

Table 1: AR Estimations

	AR1	AR2	AR3
ar1	0.9983	1.0884	1.0919
	(0.0020)	(0.0504)	(0.0506)
intercept	517.4887	516.8318	517.3178
	(19.5350)	(18.7930)	(19.1239)
ar2		-0.0902	-0.1336
		(0.0505)	(0.0746)
ar3			0.0399
			(0.0506)
nobs	390	390	390
sigma	1.2932	1.2880	1.2869
logLik	-656.5286	-654.9411	-654.6302
AIC	1319.0572	1317.8822	1319.2604
BIC	1330.9556	1333.7468	1339.0912
nobs.1	390.0000	390.0000	390.0000

*** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

```
table2 = export_summs(iidcheck1,iidcheck2,iidcheck3,
             model.names = c("AR1 Residuals","AR2 Residuals","AR3 Residuals"),
             digits = 4)
huxtable::caption(table2) <- "Checking Residuals"
huxtable::set_width(table2, 0.8)
```

Volatility

JPR Formula

$$v_t = \frac{1}{N}\sum_{i=1}^{N}(\Delta p_{t,i})^2$$

where Δp_t is the difference in price (open - close)
and i represents every minute

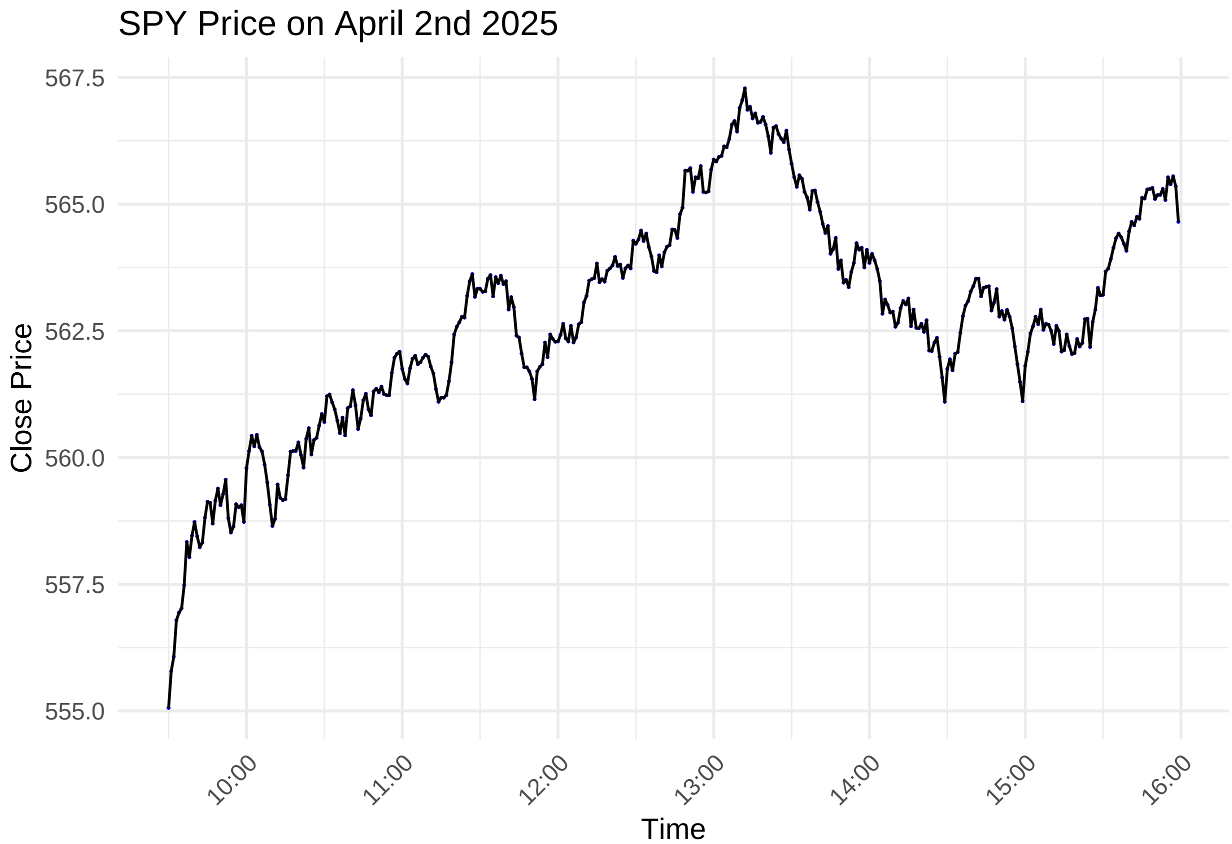
```
#extract a particular day
day_SPY_0402 = day_selector(raw_SPY,2025,04,02) #april 2nd 2025
```

Table 2: Checking Residuals

	AR1 Residuals	AR2 Residuals	AR3 Residuals
(Intercept)	0.1102	0.1092	0.1135
	(0.0655)	(0.0655)	(0.0654)
AR1res_lagged	0.0799		
	(0.0506)		
AR2res_lagged		-0.0054	
		(0.0508)	
AR3res_lagged			-0.0078
			(0.0508)
N	389	389	389
R2	0.0064	0.0000	0.0001

*** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$.

```r
#let's plot it
day_plotter(day_SPY_0402,"SPY Price on April 2nd 2025")
```




```r
#quick
quickplot = function(x){day_plotter(day_selector(raw_SPY,2025,04,x),"SPY Price 2025")}
quickplot(9)
```


```
#realized volatility
delta_price = day_SPY_0402$close - day_SPY_0402$open
delta_price_sqr = delta_price^2
day_SPY_0402 = cbind(day_selector(raw_SPY,2025,04,02),delta_price_sqr)
v_t = sum(delta_price_sqr) / length(delta_price)

#or is it like this??
#realized volatility method2
p_t = day_SPY_0402$close
p_t_1 <- lag(p_t, 1)
delta_price2 = p_t_1 - p_t
v_t2 = sum((na.omit(delta_price2))^2) / length(na.omit(delta_price2))
```

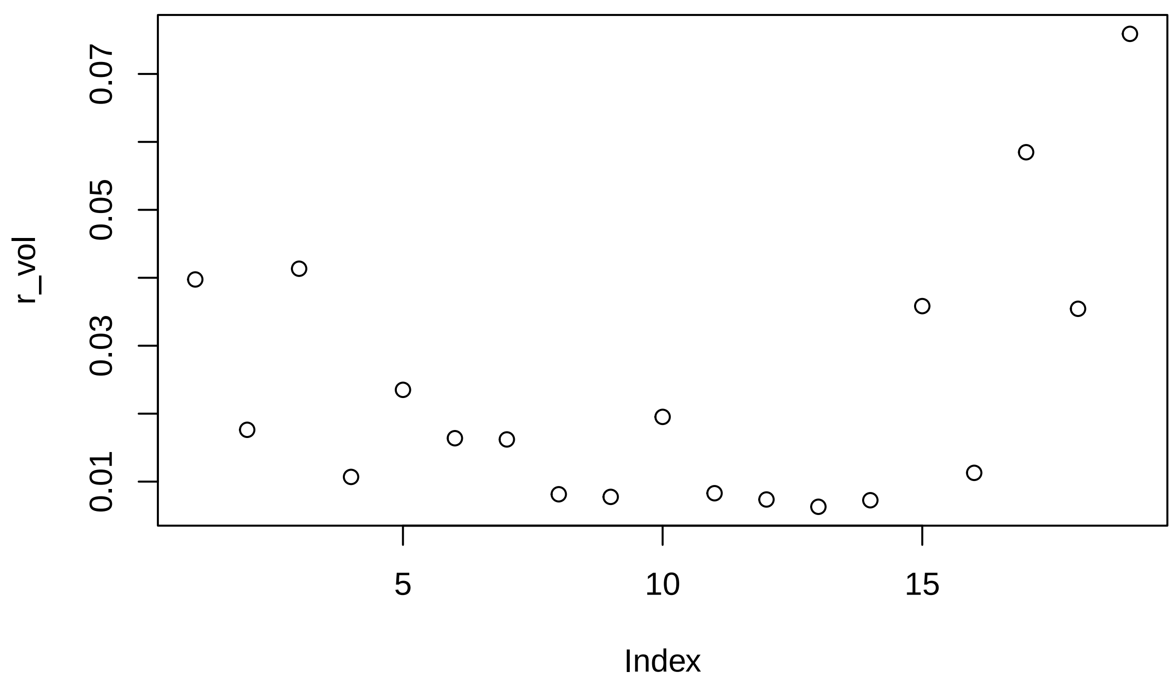
```
#simplify notation
data = raw_SPY2021_01

#find all days excluding weekends etc
days = unique(format(as.Date(data$timestamp,
                      format = "%Y-%m-%d %H:%M:%S"),format="%d"))
months = unique(format(as.Date(data$timestamp,
                      format = "%Y-%m-%d %H:%M:%S"),format="%m"))
years = unique(format(as.Date(data$timestamp,
                      format = "%Y-%m-%d %H:%M:%S"),format="%Y"))
r_vol = c(1:length(days))

for (i in 1:length(days)){
```

```
  daydata = day_selector(data,2021,01,as.numeric(days[i])) #selects data for each day
  r_vol[i] = r.vol(daydata) #computes realized volatility for each day
  r_vol
}

plot(r_vol)
```



Garman and Klass (1980) Formula

Note that this formula uses open-high-low-close information. \ This model is based on the assumption that price returns follow a Wiener process with zero drift and constant infinitesimal variance. It's constructed by minimizing the variance of a quadratic estimator subject to the constraints of price and time symmetry and scale invariance of volatility. Source: https://assets.bbhub.io/professional/sites/10/intraday_volatility-3.pdf

$$V_{ohlc} = 0.5[log(H) - log(L)]^2 - [2log(2) - 1][log(C) - log(O)]^2$$

```
#extract a particular day
day_SPY_0402 = day_selector(raw_SPY,2025,04,02) #april 2nd 2025

#variables
C = day_SPY_0402$close
O = day_SPY_0402$open
H = day_SPY_0402$high
L = day_SPY_0402$low
```

```r
#realized volatility
V_ohlc = 0.5*(log(H)-log(L))^2 - (2*log(2)-1)*(log(C)-log(O))^2
v_ohlc = sqrt(V_ohlc)
day_SPY_0402 = cbind(day_selector(raw_SPY,2025,04,02),V_ohlc)
avg = sum(V_ohlc) / length(V_ohlc)
```