Load R Packages

```
install.packages(c("dplyr", "dtplyr", "data.table", "lubridate",
"ggplot2", "PerformanceAnalytics", "xts"))
```

```
The downloaded binary packages are in
    /var/folders/by/g895l7l128j9y19qqn66xcxh0000gn/T//RtmpHEdE2f/downloaded_packages
```

```
library(dplyr)
library(dtplyr)
library(data.table)
library(lubridate)
library(ggplot2)
library(PerformanceAnalytics)
library(xts)

options(repr.plot.width = 10, repr.plot.height = 5)  # For Jupyter notebooks
```

```
df <- fread("/Users/ivanhung/Documents/GitHub/final-r-assignment/dataset.csv")
```

Cleaning dataset

```
# Setting datadate to a date object
df %>% mutate(datadate = as.Date(datadate, format = "%m/%d/%Y")) -> df

# Remove irrelevant columns and other stocks except for Pfizer (PFE)
pfe <- df[tic == "PFE",
          .(datadate, cshtrd, prccd, prchd, prcld, prcod)]
```

```r
head(pfe)

# Plot time series of PFE's closing prices
ggplot(data = pfe, aes(x = datadate, y = prccd)) +
  geom_line() +
  labs(title = "Pfizer (PFE) Closing Stock Prices Over Time",
       x = "Date",
       y = "Closing Price (USD)") +
  theme_minimal()
```

A data.table: 6 × 6

datadate <date>	cshtrd <dbl>	prccd <dbl>	prchd <dbl>	prcld <dbl>	prcod <dbl>
2010-01-04	52074710	18.93	18.94	18.235	18.27
2010-01-05	43368460	18.66	18.93	18.550	18.92
2010-01-06	41405070	18.60	18.81	18.510	18.66
2010-01-07	39427720	18.53	18.67	18.460	18.64
2010-01-08	30403370	18.68	18.71	18.520	18.62
2010-01-11	32442710	18.83	18.95	18.670	18.83

Next, calculate simple returns so that we can make our prices stationary and allow us to have a better understanding our data as we can proceed to plot Pfizer's ACF and PACF plots and confirm for certain statistical characteristics.

```
# Calculating simple returns (simple returns in %),
# which we will denote as s_ret (s_ret_per)

pfe <- pfe %>%
    mutate(ret = round(((prccd/lag(prccd))-1), 4)) %>%
    mutate(s_ret = round(((prccd/lag(prccd))-1)*100, 4))

head(pfe)
```

A data.table: 6 × 8

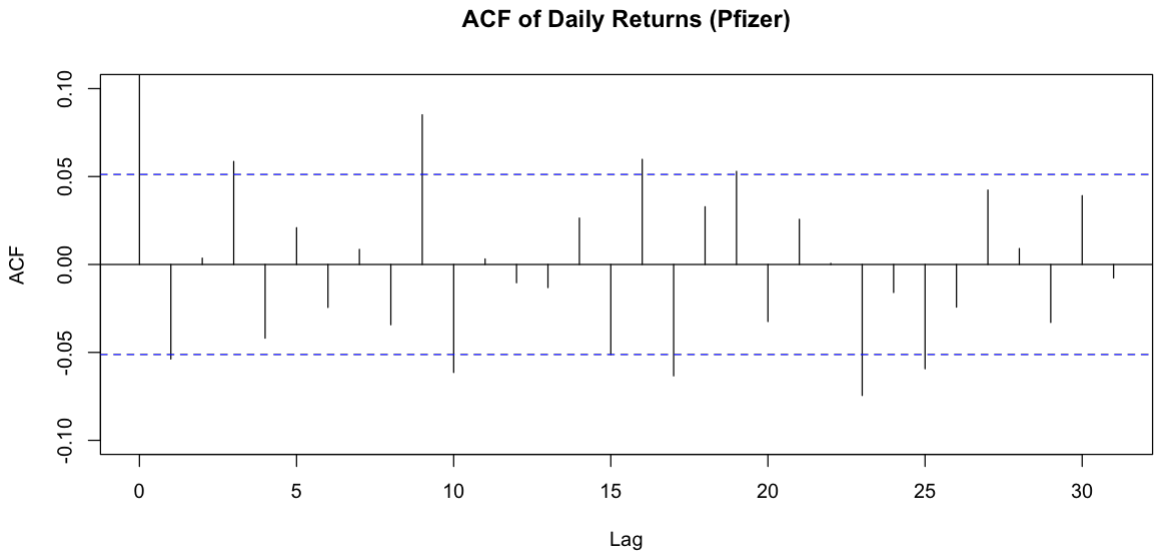
datadate <date>	cshtrd <dbl>	prccd <dbl>	prchd <dbl>	prcld <dbl>	prcod <dbl>	ret <dbl>	s_ret <dbl>
2010-01-04	52074710	18.93	18.94	18.235	18.27	NA	NA
2010-01-05	43368460	18.66	18.93	18.550	18.92	-0.0143	-1.4263
2010-01-06	41405070	18.60	18.81	18.510	18.66	-0.0032	-0.3215
2010-01-07	39427720	18.53	18.67	18.460	18.64	-0.0038	-0.3763
2010-01-08	30403370	18.68	18.71	18.520	18.62	0.0081	0.8095
2010-01-11	32442710	18.83	18.95	18.670	18.83	0.0080	0.8030

Next, we should plot the autocorrelation and partial-autocorrelation functions of our closing prices to identify if there are any seasonal structures or autocorrelation that we might need to deal with.

```
# ACF plot to identify any autocorrelation or seasonality patterns in our data
#| warning: false
#| results: hide
acf(pfe$ret, lags = 20,
    na.action = na.omit,
    main = "ACF of Daily Returns (Pfizer)",
    ylim = c(-0.1,0.1))
```

```
Warning message in plot.window(...):
""lags" is not a graphical parameter"
```

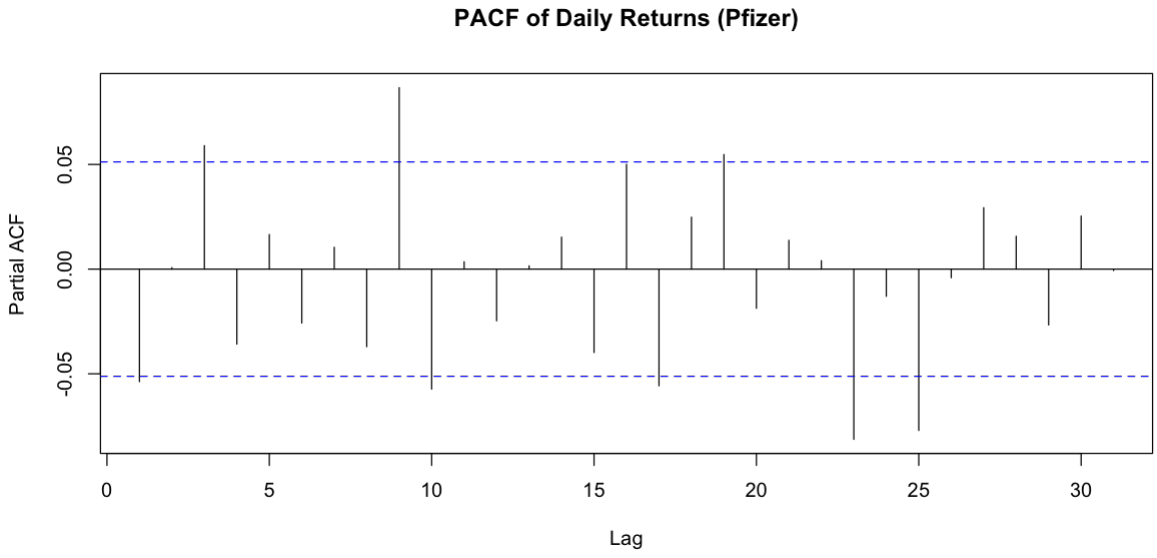
```
Warning message in plot.xy(xy, type, ...):
""lags" is not a graphical parameter"
Warning message in axis(side = side, at = at, labels = labels, ...):
""lags" is not a graphical parameter"
Warning message in axis(side = side, at = at, labels = labels, ...):
""lags" is not a graphical parameter"
Warning message in box(...):
""lags" is not a graphical parameter"
Warning message in title(...):
""lags" is not a graphical parameter"
Warning message in plot.xy(xy, type, ...):
""lags" is not a graphical parameter"
Warning message in axis(side = side, at = at, labels = labels, ...):
""lags" is not a graphical parameter"
Warning message in axis(side = side, at = at, labels = labels, ...):
""lags" is not a graphical parameter"
Warning message in box(...):
""lags" is not a graphical parameter"
Warning message in title(...):
""lags" is not a graphical parameter"
```




```
# PACF plot
#| warning: false
#| results: hide
```

```r
pacf(pfe$ret, lags = 20,
    na.action = na.omit,
    main = "PACF of Daily Returns (Pfizer)")
```

```
Warning message in plot.window(...):
""lags" is not a graphical parameter"
Warning message in plot.xy(xy, type, ...):
""lags" is not a graphical parameter"
Warning message in axis(side = side, at = at, labels = labels, ...):
""lags" is not a graphical parameter"
Warning message in plot.xy(xy, type, ...):
""lags" is not a graphical parameter"
Warning message in axis(side = side, at = at, labels = labels, ...):
""lags" is not a graphical parameter"
Warning message in axis(side = side, at = at, labels = labels, ...):
""lags" is not a graphical parameter"
Warning message in box(...):
""lags" is not a graphical parameter"
Warning message in title(...):
""lags" is not a graphical parameter"
Warning message in box(...):
""lags" is not a graphical parameter"
Warning message in title(...):
""lags" is not a graphical parameter"
```

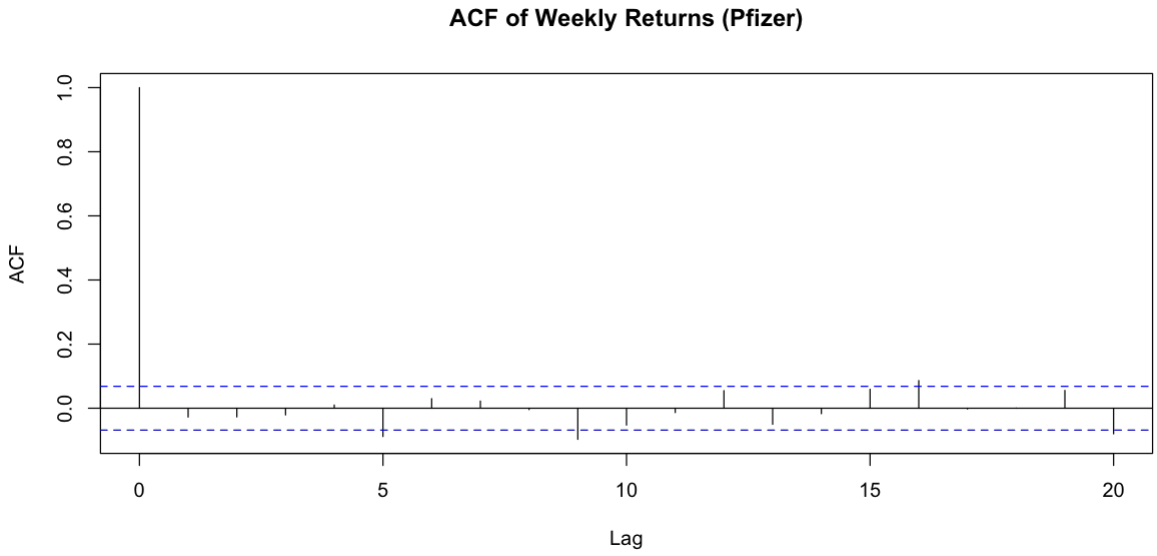


We can focus in on a specific time horizon and see if autocorrelation exists within a certain timeframe. This evidence can help us to determine whether there is statistical arbitrage in which our trading strategy can detect a pattern can profit from (potentially short term momentum), as opposed to having white noise (returns which follow a strong form of the EMH where all information about the stock is reflected in its prices). We decide to focus on a relatively long horizon as it can provide us more information of how the stock's price changed before, during, and after COVID; providing us with a holistic story and finding opportunities for statistical arbitrage in the '3' phases of COVID.

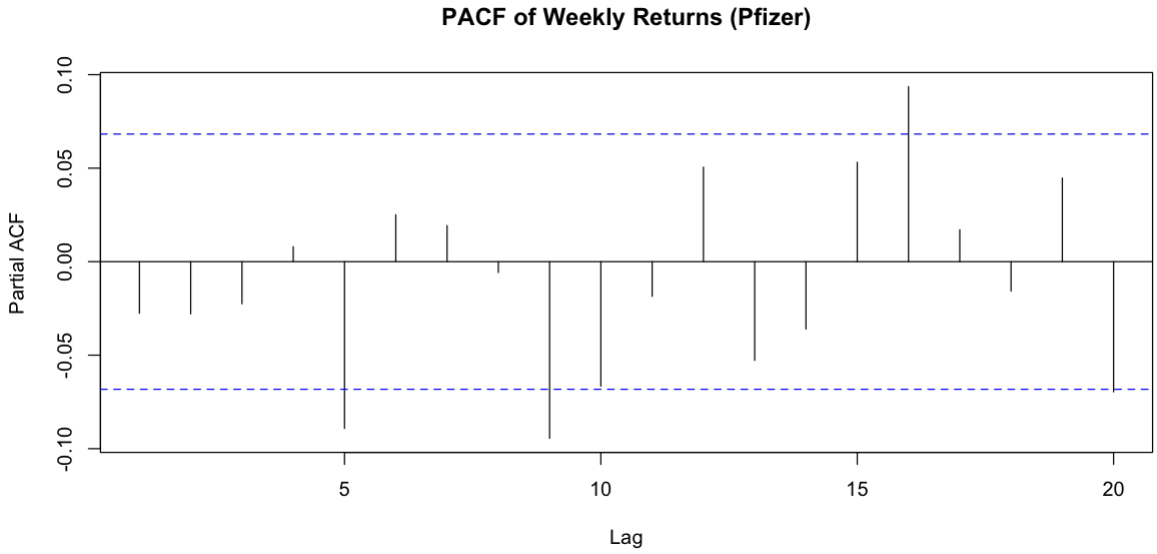
```
# Create a list of periods we want to look at
periods <- list(
    weekly = "week",
    monthly = "month",
    semi_annually = "6 months",
    annually = "year"
)

# Loop through each time period
for (period_name in names(periods)) {
    cat("Processing:", toupper(period_name), "\n")

    # Create proper period returns (one observation per period)
    pfe_period <- pfe %>%
        mutate(period = floor_date(datadate, periods[[period_name]])) %>%
        group_by(period) %>%
        arrange(datadate) %>%
        slice_tail(n = 1) %>%
        ungroup() %>%
        arrange(datadate) %>%
        mutate(period_ret = (prccd / lag(prccd)) - 1) %>%
        filter(!is.na(period_ret))

    # View tibble (dataframe) using the following:
    # print(head(pfe_period))
    # cat("Number of periods:", nrow(pfe_period), "\n\n")

    # ACF of actual period returns (suppress warnings)
    suppressWarnings(
        acf(pfe_period$period_ret, lag.max = 20,
            na.action = na.omit,
            main = paste("ACF of",
            tools::toTitleCase(gsub("_", " ", period_name)),
            "Returns (Pfizer)"))
```

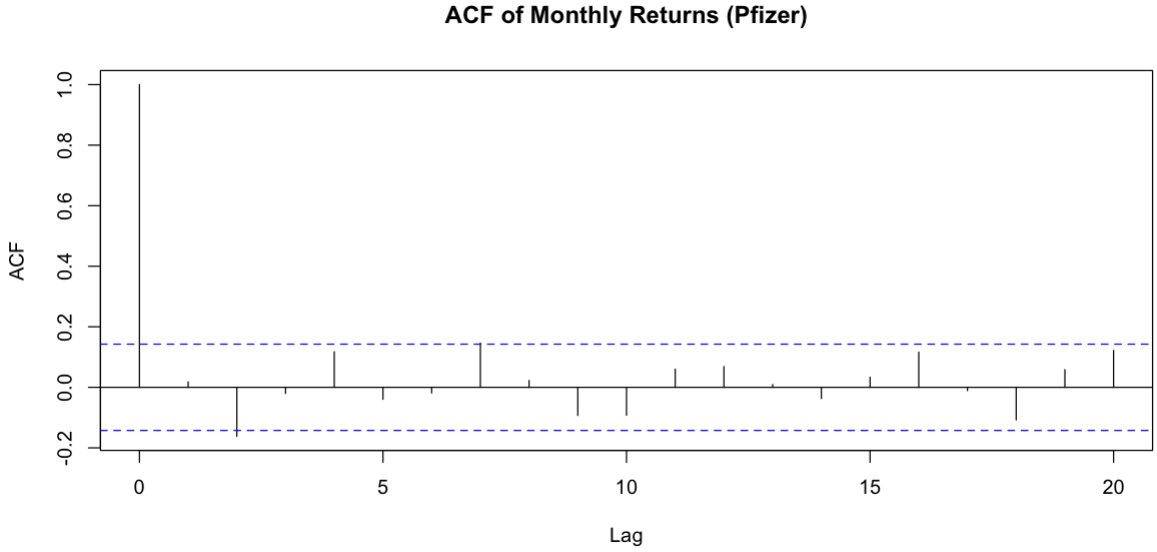
```
    )

    # PACF of actual period returns (suppress warnings)
    suppressWarnings(
        pacf(pfe_period$period_ret, lag.max = 20,
             na.action = na.omit,
             main = paste("PACF of",
             tools::toTitleCase(gsub("_", " ", period_name)),
             "Returns (Pfizer)"))
    )
}
```

```
Processing: WEEKLY
```

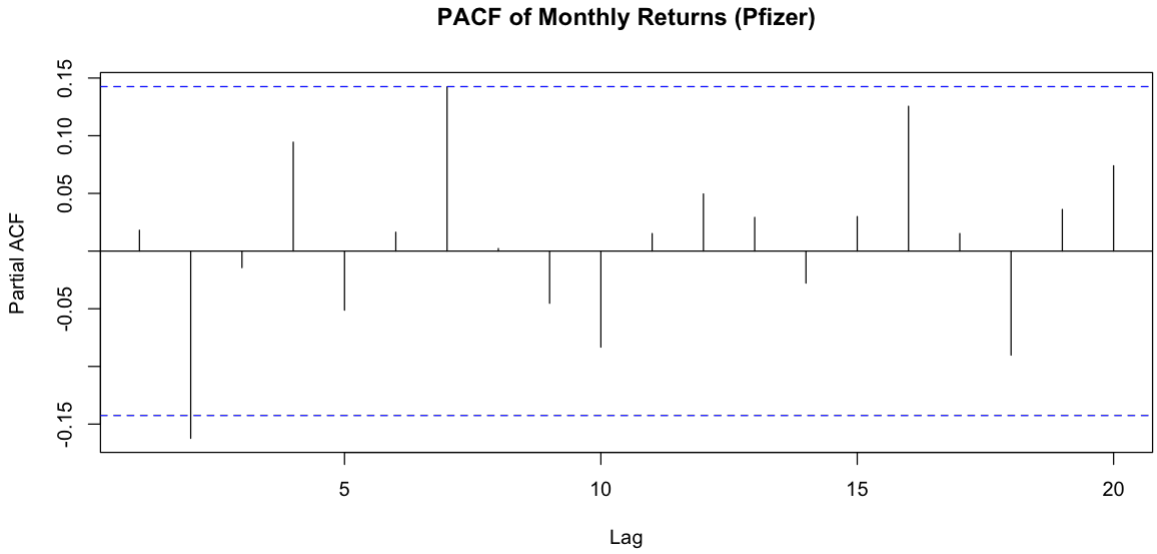


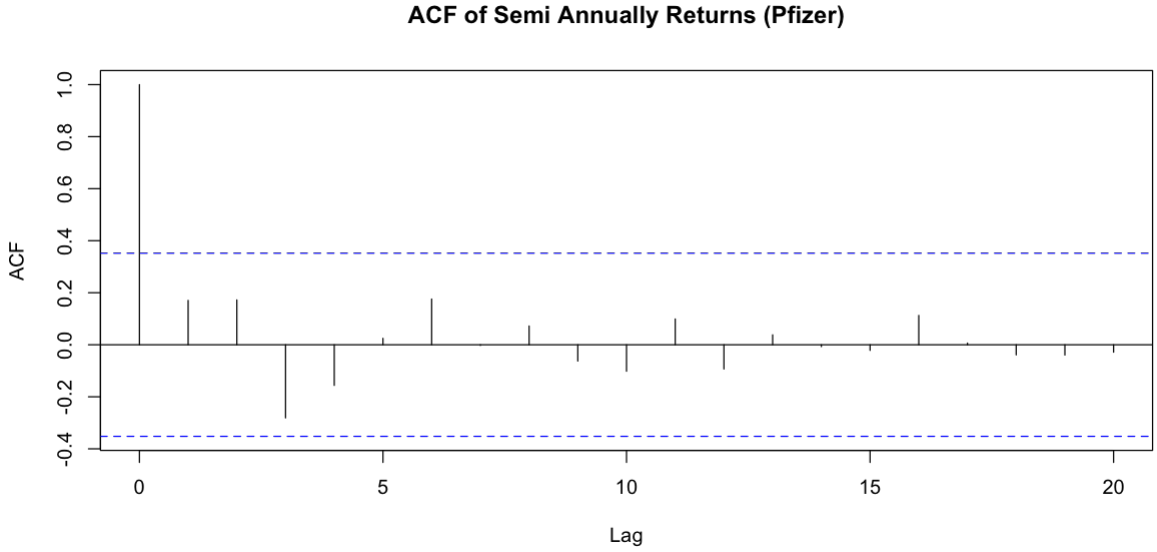

```
Processing: MONTHLY
```



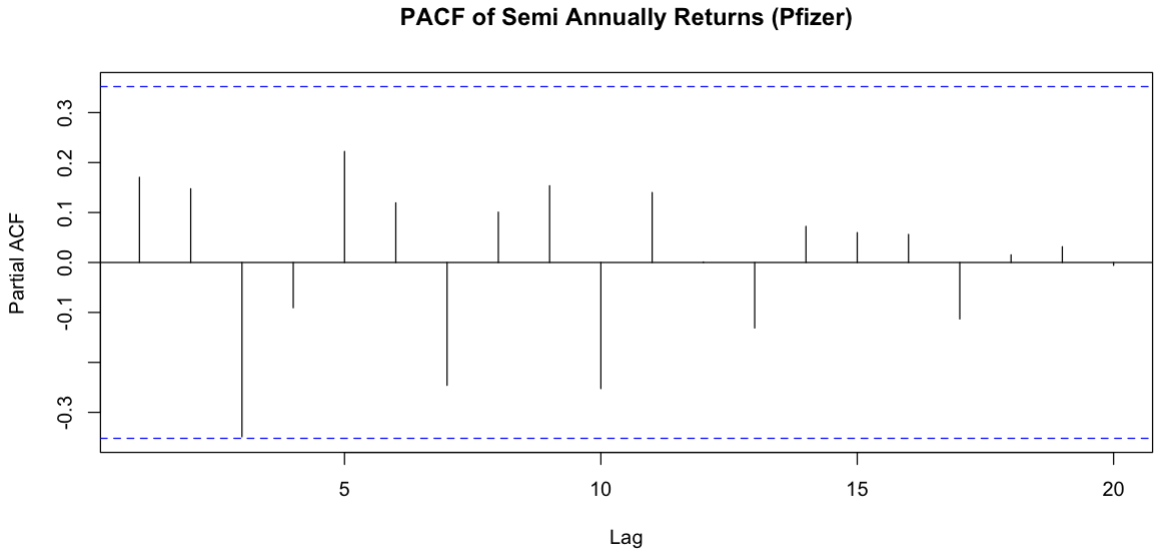


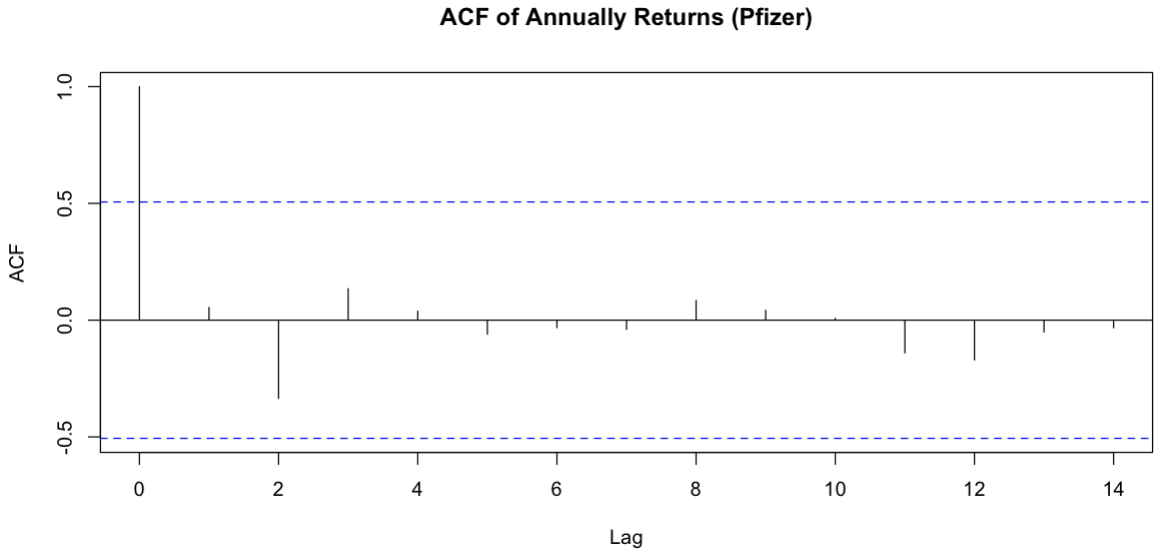

```
Processing: SEMI_ANNUALLY
```

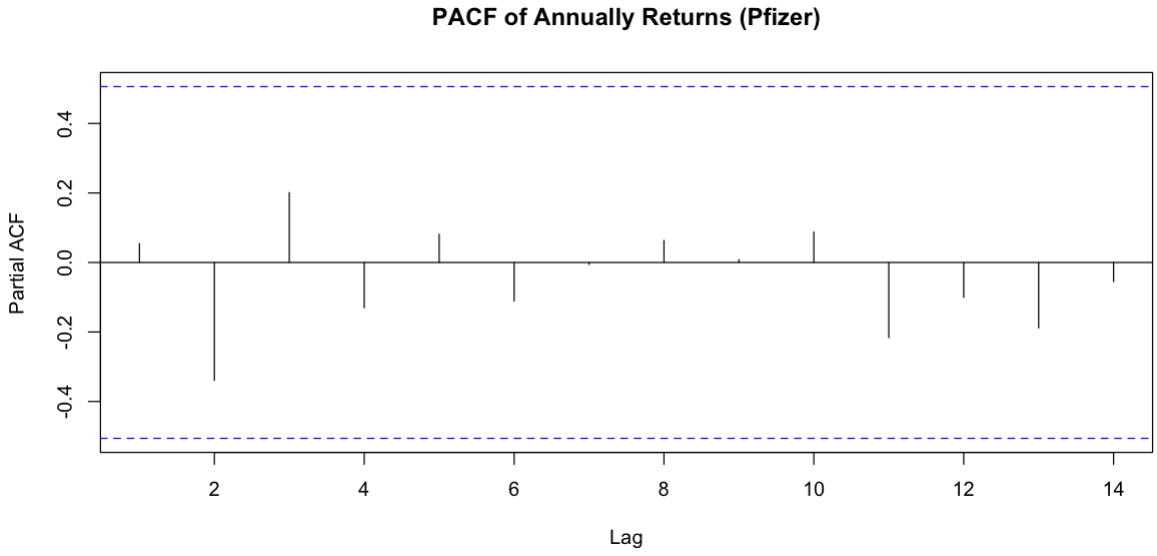





```
Processing: ANNUALLY
```

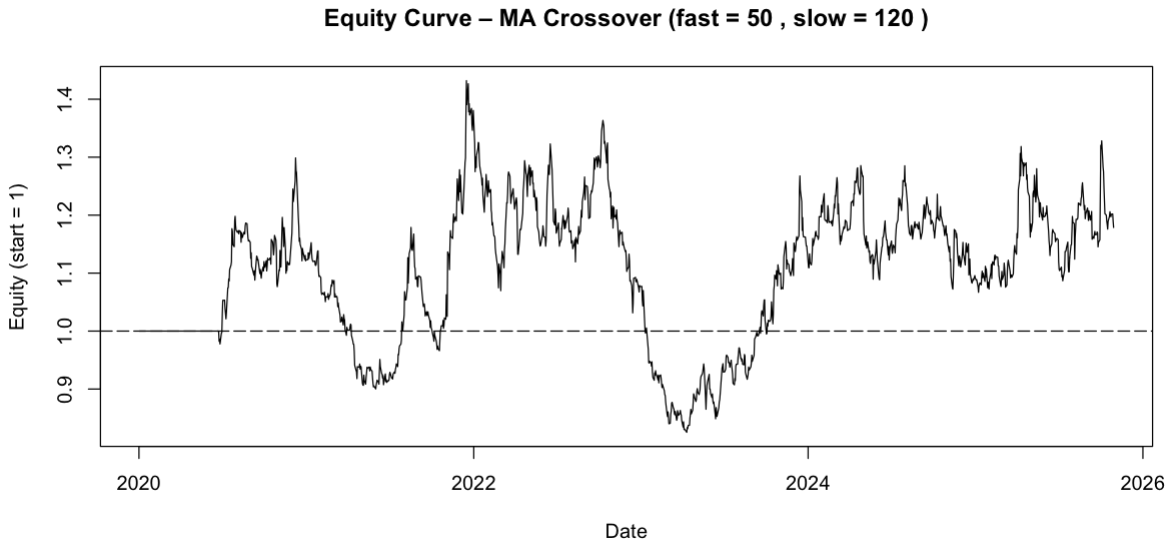

PACF of Semi Annually Returns (Pfizer)
Partial ACF
0.3
0.2
0.1
0.0
-0.1
-0.3
5
10
15
20
Lag


ACF of Annually Returns (Pfizer)
ACF
1.0
0.5
0.0
-0.5
0
2
4
6
8
10
12
14
Lag

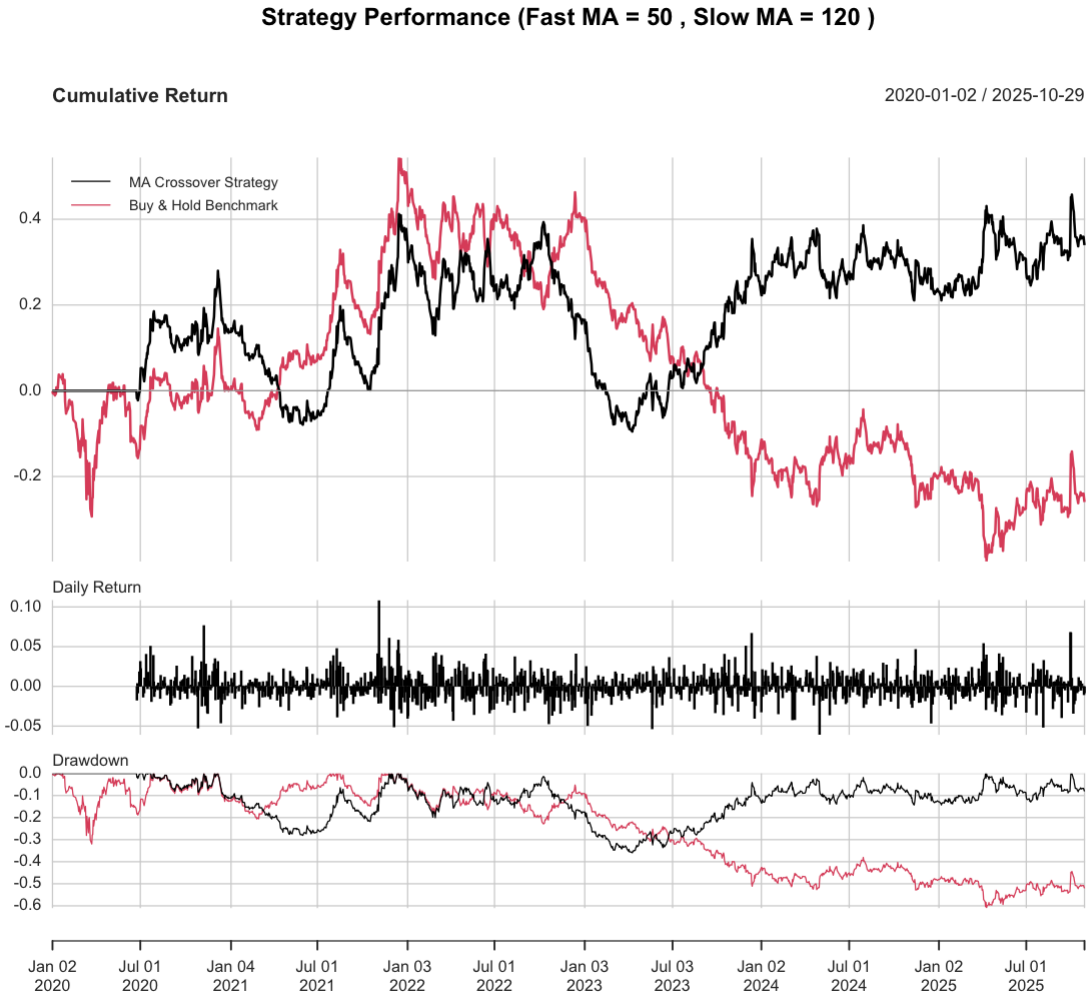



```
pfe <- pfe %>%
  filter(datadate >= as.Date("2020-01-01")) %>%
  arrange(datadate)
```

```
price <- pfe$prccd

# Generate signals and backtest using the optimal windows
best_signal <- ma_signal(price, fast = best_fast, slow = best_slow)
best_bt     <- backtest_ma(price, signal = best_signal)

# Equity curve plot
plot(
  pfe$datadate, best_bt$equity, type = "l",
  xlab = "Date", ylab = "Equity (start = 1)",
  main = paste("Equity Curve - MA Crossover (fast =", best_fast,
               ", slow =", best_slow, ")")
)
abline(h = 1, lty = 5)
```

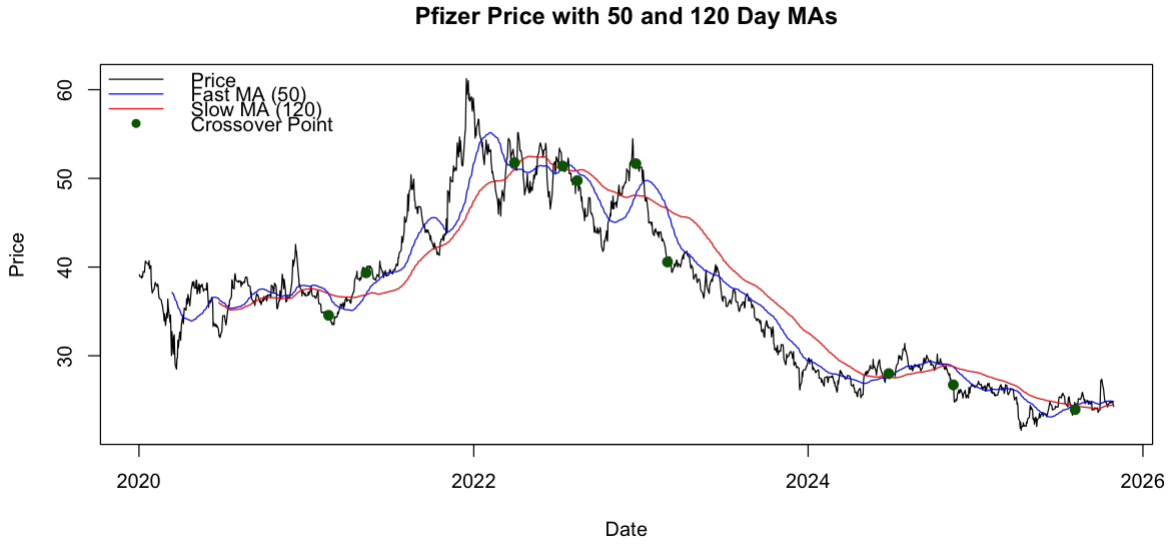



```r
# Store data in an xts readable dataframe
strategy_returns <- xts(best_bt$returns, order.by = pfe$datadate)
benchmark_returns <- xts(pfe$ret, order.by = pfe$datadate)

colnames(strategy_returns) <- ("MA Crossover Strategy (50,120)")
colnames(benchmark_returns) <- ("Buy & Hold Benchmark (PFE Returns)")

# Performance Summary Chart
options(repr.plot.width = 10, repr.plot.height = 9) # Adjust plot dim.
charts.PerformanceSummary(
    comparison,
    geometric = FALSE,
    main = paste("Strategy Performance (Fast MA =",
        best_fast, ", Slow MA =", best_slow, ")")
)

# Reset to default height
options(repr.plot.width = 10, repr.plot.height = 5)
```




```r
# Compute moving averages for the best windows
fast_ma_best <- moving_avg(price, win_size = best_fast)
slow_ma_best <- moving_avg(price, win_size = best_slow)

# Base plot: price
plot(
  pfe$datadate, price, type = "l",
  xlab = "Date", ylab = "Price",
  main = paste("Pfizer Price with", best_fast,
  "and", best_slow, "Day MAs")
)
```

```r
# Add moving averages
lines(pfe$datadate, fast_ma_best, col = "blue")
lines(pfe$datadate, slow_ma_best, col = "red")

# Identify crossover points
spread    <- fast_ma_best - slow_ma_best
cross_idx <- which(diff(sign(spread)) != 0) + 1

# Add crossover markers to the plot
points(
  pfe$datadate[cross_idx],
  price[cross_idx],
  pch = 16, col = "darkgreen", cex = 1.2
)

legend(
  "topleft",
  legend = c(
    "Price",
    paste0("Fast MA (", best_fast, ")"),
    paste0("Slow MA (", best_slow, ")"),
    "Crossover Point"
  ),
  col = c("black", "blue", "red", "darkgreen"),
  lty = c(1, 1, 1, NA),
  pch = c(NA, NA, NA, 16),
  bty = "n"
)
```


Pfizer Price with 50 and 120 Day MAs
Price
Fast MA (50)
Slow MA (120)
Crossover Point
Price
30
40
50
60
2020
2022
2024
2026
Date