

# Recruiting Challenge

## Performance of the S&P500 in 2019

Shukry Zablah

## Contents

<b>Data Wrangling</b>	<b>1</b>
Data Retrieval and Cleanup . . . . .	1
Sector Performance . . . . .	3
Relative Annual Returns . . . . .	4
<b>Results</b>	<b>4</b>
Sector Performance in 2019 . . . . .	4
Relative Annual Returns in 2019 . . . . .	5

```
library(tidyverse)
library(rvest)
library(janitor)
library(tidyquant)
```

```
parse_human_readable_number <- function(repr) {
  if(is.na(repr)) {
    return(NA_real_)
  }
  if(str_detect(repr, "B")) {
    return(1e9 * parse_number(repr))
  } else if(str_detect(repr, "M")) {
    return(1e6 * parse_number(repr))
  }
}
```

```
weighted_mean <- function(x, w, ..., na.rm=F) {
  if(na.rm) {
    x1 <- x[!is.na(x)&!is.na(w)]
    w <- w[!is.na(x)&!is.na(w)]
    x <- x1
  }
  weighted.mean(x, w, ..., na.rm=F)
}
```

## Data Wrangling

### Data Retrieval and Cleanup

The first goal is to retrieve the list of companies that are currently listed in the S&P500 and their corresponding GICS sectors. We also retrieve the latest information for these stocks – such as market capitalization – from

the different stock exchanges.

```
sp500_companies <- tq_index("SP500") %>%
  left_join(bind_rows(tq_exchange("AMEX"),
                      tq_exchange("NASDAQ"),
                      tq_exchange("NYSE")),
            by = "symbol") %>%
  distinct() %>%
  select(symbol, company = company.x, weight, sector = sector.x,
         shares_held, market_cap = market.cap, industry)

sp500_companies %>% glimpse()
```

```
## Observations: 505
## Variables: 7
## $ symbol      <chr> "AAPL", "MSFT", "AMZN", "FB", "BRK.B", "GOOGL", "GOOG",...
## $ company     <chr> "Apple Inc.", "Microsoft Corporation", "Amazon.com Inc....
## $ weight      <dbl> 0.047931162, 0.045515896, 0.029004745, 0.019405346, 0.0...
## $ sector      <chr> "Information Technology", "Information Technology", "Co...
## $ shares_held <dbl> 48669720, 88896700, 4853403, 28042064, 22793356, 349117...
## $ market_cap  <chr> "$1389.56B", "$1245.63B", "$937.7B", "$632.83B", NA, "$...
## $ industry    <chr> "Computer Manufacturing", "Computer Software: Prepackag...
```

Now that we have data for all 505 stocks in the exchange, their weights, and their market capitalization, we query the Yahoo Finance API for historical trading data. We only need data with monthly resolution from 2018-12-31 to 2019-12-31.

```
sp500_raw <- sp500_companies %>%
  pull(symbol) %>%
  map_chr(~ str_replace(.x, "\\.", "-")) %>% ## cleanup tickers for API call
  tidycorpus::tq_get(from = "2018-12-31", ## defaults to Yahoo Finance API
                    to = "2020-01-01") %>%
  mutate(symbol = map_chr(symbol, ~ str_replace(.x, "-", "\\."))) ## cleanup

sp500_raw %>% glimpse()
```

```
## Observations: 127,468
## Variables: 8
## $ symbol      <chr> "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "AAPL", "A...
## $ date        <date> 2018-12-31, 2019-01-02, 2019-01-03, 2019-01-04, 2019-01-0...
## $ open        <dbl> 158.53, 154.89, 143.98, 144.53, 148.70, 149.56, 151.29, 15...
## $ high        <dbl> 159.36, 158.85, 145.72, 148.55, 148.83, 151.82, 154.53, 15...
## $ low         <dbl> 156.48, 154.23, 142.00, 143.80, 145.90, 148.52, 149.63, 15...
## $ close       <dbl> 157.74, 157.92, 142.19, 148.26, 147.93, 150.75, 153.31, 15...
## $ volume      <dbl> 35003500, 37039700, 91312200, 58607100, 54777800, 41025300...
## $ adjusted    <dbl> 155.4050, 155.5824, 140.0852, 146.0654, 145.7403, 148.5185...
```

The raw data contains the usual trading information. We will focus on the adjusted close values (which include dividends, splits, etc). Now we perform a preprocessing step to make computations on the performance of assets more straightforward.

```
start <- sp500_raw %>% filter(date == "2018-12-31")
end <- sp500_raw %>% filter(date == "2019-12-31")

sp500_processed <- start %>%
  full_join(end,
            by = "symbol",
```

```

suffix = c("_2018", "_2019")) %>%
left_join(sp500_companies, by = "symbol") %>%
mutate(return = (adjusted_2019 - adjusted_2018) * 100 / adjusted_2018)

```

```
sp500_processed
```

```

## # A tibble: 505 x 22
##   symbol date_2018 open_2018 high_2018 low_2018 close_2018 volume_2018
##   <chr>   <date>      <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 AAPL   2018-12-31    159.    159.    156.    158.    35003500
## 2 MSFT   2018-12-31    101.    102.    100.    102.    33173800
## 3 AMZN   2018-12-31   1511.   1521.   1487    1502.    6954500
## 4 FB     2018-12-31    134.    135.    130.    131.    24625300
## 5 BRK.B  2018-12-31    204.    205.    201.    204.    5451900
## 6 GOOGL  2018-12-31   1058.   1063.   1033.   1045.    1655500
## 7 GOOG   2018-12-31   1051.   1053.   1024.   1036.    1493300
## 8 JPM    2018-12-31    97.6    98.8    96.8    97.6    13237200
## 9 JNJ    2018-12-31    128.    130.    127.    129.    7409900
## 10 V     2018-12-31    132.    132.    130.    132.    7976000
## # ... with 495 more rows, and 15 more variables: adjusted_2018 <dbl>,
## #   date_2019 <date>, open_2019 <dbl>, high_2019 <dbl>, low_2019 <dbl>,
## #   close_2019 <dbl>, volume_2019 <dbl>, adjusted_2019 <dbl>, company <chr>,
## #   weight <dbl>, sector <chr>, shares_held <dbl>, market_cap <chr>,
## #   industry <chr>, return <dbl>

```

Further computations on the 2019 performance are straightforward as the return is already computed for all 505 stocks in the S&P500.

## Sector Performance

Calculating the weighted mean is straightforward after our preprocessing step. We are computing, for each sector, the average weighted by the market capitalization of the companies.

```

sector_performance <- sp500_processed %>%
  mutate(market_cap = map_dbl(market_cap, parse_human_readable_number)) %>%
  group_by(sector) %>%
  summarize(wt_avg_return = weighted_mean(return, market_cap, na.rm=T)) %>%
  arrange(desc(wt_avg_return))

```

```
sector_performance
```

```

## # A tibble: 11 x 2
##   sector          wt_avg_return
##   <chr>          <dbl>
## 1 Information Technology    55.7
## 2 Financials                37.2
## 3 Communication Services   34.2
## 4 Industrials              32.7
## 5 Real Estate              32.7
## 6 Materials                31.1
## 7 Consumer Discretionary    30.4
## 8 Consumer Staples         29.8
## 9 Utilities                29.8
## 10 Health Care              22.5
## 11 Energy                  12.9

```

The result shows the weight average return for the eleven sectors ranked in descending order.

## Relative Annual Returns

To find the relative annual return for each of the 505 stocks, we join the previous computation of returns in our processed data with the sector performance, and perform the simple vectorized computation.

```
rel_annual_returns <- sp500_processed %>%
  left_join(sector_performance, by = "sector") %>%
  mutate(rel_return = return / wt_avg_return) %>%
  select(symbol, sector, rel_return) %>%
  arrange(desc(rel_return))
```

```
rel_annual_returns
```

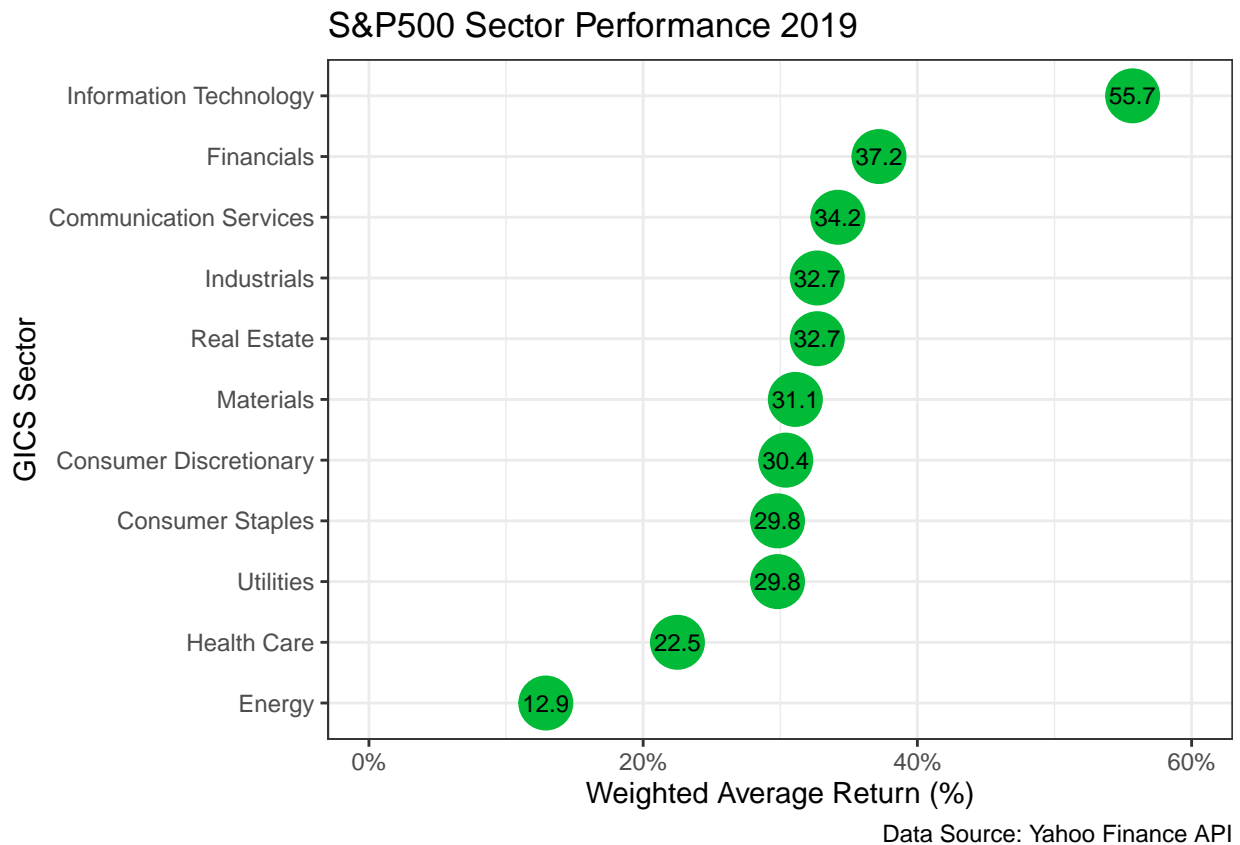
```
## # A tibble: 505 x 3
##   symbol sector      rel_return
##   <chr>  <chr>      <dbl>
## 1 HES    Energy      5.25
## 2 OKE    Energy      3.70
## 3 KMI    Energy      3.44
## 4 TGT    Consumer Discretionary 3.30
## 5 CMG    Consumer Discretionary 3.09
## 6 CPRT   Industrials    2.76
## 7 NBL    Energy      2.74
## 8 AMD    Information Technology 2.66
## 9 COTY    Consumer Staples 2.65
## 10 PSX   Energy      2.63
## # ... with 495 more rows
```

The table above contains the ratio of annual return over respective sector performance for 2019 for all 505 stocks.

## Results

### Sector Performance in 2019

```
sector_performance %>%
  mutate(sector = fct_reorder(sector, wt_avg_return),
         wt_avg_return = round(wt_avg_return, 1)) %>%
  ggplot(aes(x = sector, y = wt_avg_return, label = wt_avg_return)) +
  geom_point(col="#00ba38", size=9) + # Draw points
  geom_text(color="black", size=3) +
  geom_vline(color="grey", linetype="dashed", size = 2, xintercept=28.8) +
  scale_y_continuous(limits = c(0, 60),
                    labels = scales::label_percent(scale=1)) +
  labs(title="S&P500 Sector Performance 2019",
       caption="Data Source: Yahoo Finance API",
       x = "GICS Sector",
       y = "Weighted Average Return (%)") +
  coord_flip() +
  theme_bw()
```

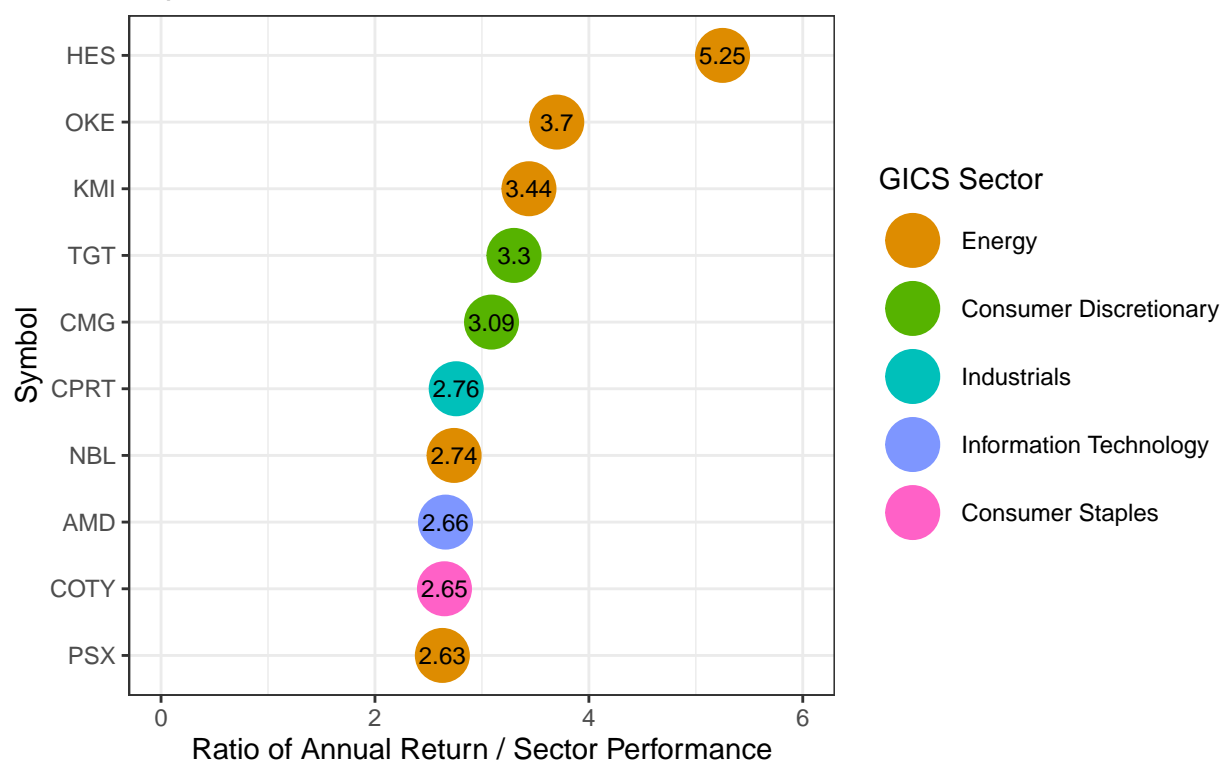


```
ggsave("sectors.png", width = 6.5, height = 4.3)
```

### Relative Annual Returns in 2019

```
rel_annual_returns %>%
  arrange(desc(abs(rel_return))) %>%
  head(10) %>%
  mutate(symbol = fct_reorder(symbol, rel_return),
         rel_return = round(rel_return, 2),
         sector = fct_reorder(sector, rel_return, .desc=T)) %>%
  ggplot(aes(x = symbol, y = rel_return, label = rel_return)) +
  geom_point(aes(col = sector), size = 9) +
  scale_color_hue(h.start=30) +
  geom_text(color="black", size=3) +
  scale_y_continuous(limits = c(0, 6)) +
  labs(title="Top 10 Stock Movements Relative To Their Sector",
       caption="Data Source: Yahoo Finance API",
       x = "Symbol",
       y = "Ratio of Annual Return / Sector Performance",
       color = "GICS Sector") +
  coord_flip() +
  theme_bw()
```

## Top 10 Stock Movements Relative To Their Sector



Data Source: Yahoo Finance API

```
ggsave("relreturn.png", width = 6.5, height = 4.3)
```