Final_Walkthrough_FinalProject

Adam Johnson, Nick Loeffelholz, Christopher Raddatz, Yatri Patel

2024-04-21

Introduction

This RMarkdown document will serve as a code walk through for most of the data collection, cleaning and analytical aspects of our project we did in R. The following segments will be broken down in that manner:
1) Data Collection & Cleaning
2) Model Building
3) Additional Filtering
3a) Age Filtering
3b) Party Filtering

Packages installation

```
if (!require(tidyverse)) install.packages("tidyverse")
if (!require(tidyquant)) install.packages("tidyquant")
if (!require(PerformanceAnalytics)) install.packages("PerformanceAnalytics")
if (!require(xts)) install.packages("xts")
if (!require(lubridate)) install.packages("lubridate")
if (!require(Quandl)) install.packages("Quandl")
if (!require(riingo)) install.packages("riingo")
if (!require(eeptools)) install.packages("eeptools")
if (!require(ggplot2)) install.packages("ggplot2")
if (!require(purrr)) install.packages("purrr")
if (!require(broom)) install.packages("broom")
if (!require(maps)) install.packages("maps")
if (!require(ggthemes)) install.packages("ggthemes")
if (!require(mapproj)) install.packages("mapproj")

library(tidyverse)
library(tidyquant)
library(PerformanceAnalytics)
library(xts)
library(lubridate)
library(Quandl)
library(riingo)
library(httr)
library(tidyr)
library(dplyr)
library(jsonlite)
library(httr)
library(eeptools)
```

```
library(ggplot2)
library(purrr)
library(broom)
library(maps)
library(mapproj)
library(ggthemes)
```

Data Collection & Cleaning

We used CapitolTrades, a website that follows political activity in the stock market. Below is a function written to loop through their website, pull data as a JSON object and parse through before merging all the data together. This is an example API JSON pull, as the real data pull would take much longer. This section is an abbreviation of what we did, as we won't be saving new data and don't want to be running this script for 10+ minutes.

```
capitolTradesurl <- "https://bff.capitoltrades.com/trades"

# Make a GET request to the API
response <- GET(capitolTradesurl)

# Check if the request was successful (status code 200)
if (http_status(response)$category == "Success") {
  # Parse the JSON response
  data <- fromJSON(content(response, "text"))
} else {
  # If the request was not successful, print the error message
  print(paste("Error:", http_status(response)$reason))
}
```

```
## No encoding supplied: defaulting to UTF-8.
```

```
simpleTradeData <- data$data %>%
  select(`_txId`,txDate, txType, asset, politician, price, size, reportingGap) %>%
  unnest(c(asset, politician))

print(simpleTradeData[1:5,])
```

```
## # A tibble: 5 x 17
##       `_txId` txDate  txType assetType assetTicker instrument `_stateId` chamber
##         <dbl> <chr>   <chr>  <chr>     <chr>       <chr>      <chr>      <chr>
## 1 20003772297 2024-0~ sell   stock     NTAP:US     <NA>       co         house
## 2 20003772298 2024-0~ buy    stock     NTAP:US     <NA>       co         house
## 3 20003772299 2024-0~ sell   stock     NTAP:US     <NA>       co         house
## 4 20003772300 2024-0~ buy    stock     NTAP:US     <NA>       co         house
## 5 20003772301 2024-0~ sell   stock     NTAP:US     <NA>       co         house
## # i 9 more variables: dob <chr>, firstName <chr>, gender <chr>, lastName <chr>,
## #   nickname <chr>, party <chr>, price <dbl>, size <int>, reportingGap <int>
```

Following is the cleaning of the data that we did, so first we read the data in.

```
allCapitolTradesData <- read.csv("Data/All_CapitolTrades_Data.csv")
```

Among the financial data were tickers not listed on Yahoo Finance, to combat this we manually looked through each ticker and identified if the company had changed symbols, been delisted or were bought out.

```
update_ticker <- function(ticker_with_suffix, changed_ticker_mapping) {
  # Remove the suffix and replace "/" with "-" to match the mapping keys
  ticker_corrected <- gsub("-", "/", substr(ticker_with_suffix, 1, nchar(ticker_with_suffix) - 3))

  # If this modified ticker is in the mapping, return the new value; otherwise, return the original wit
  if(any(ticker_corrected %in% names(changed_ticker_mapping))) {
    return(changed_ticker_mapping[ticker_corrected])
  } else {
    # If not in mapping, assume the original ticker (without the last 3 characters) is still valid
    return(ticker_corrected)
  }
}
```

The function below displays our efforts to limit the stock data to stocks within the U.S and clean some of the formatting.

```
scrubCapitolTradesData <- function(capitolTradesDatadf){
  #scrub for stocks only
  capitolTradesDatadf <- capitolTradesDatadf[capitolTradesDatadf$asset.assetType == "stock",]

  #scrub for only US issues
  capitolTradesDatadf <- capitolTradesDatadf[(capitolTradesDatadf$issuer.country == "us"),]

  #change txDate to date type
  capitolTradesDatadf$txDate <- as.Date(capitolTradesDatadf$txDate, format = "%Y-%m-%d")

  #scrub to reorder
  capitolTradesDatadf <- capitolTradesDatadf[order(capitolTradesDatadf$txDate, decreasing = FALSE),]

  #some tickers have changed
  changed_ticker_mapping <- c(ORCC = "OBDC", CWEN.A = "CWEN-A", HHC = "HHH", MTBC = "CCLD",
                    MLHR = "MLKN", RLGY = "HOUS", ENOB = "RENB", VIAC = "PARA",
                    "ETWO-WS" = "ETWO", "LGF/A" = "LGF-A", "BRK/B"="BRK-B", "BF/A"="BF-A",
                    "BF/B" = "BF-B", "LGF/B" = "LGF-B", "LEN/B" = "LEN-B")

  capitolTradesDatadf$asset.assetTicker <- update_ticker(capitolTradesDatadf$asset.assetTicker, changed
  return(capitolTradesDatadf)
}
scrubbedCapitolTradesData <- scrubCapitolTradesData(allCapitolTradesData)
```

This next code block is where we collect our financial data from the stocks associated in the CapitolTrades data. We identify unique symbols, then isolate stock prices for within the dates we found in CapitolTrades. We used the tidyquant package which pulls data from Yahoo Finance to gather this data.

```
getStockPricesForTickers <- function(anyCapitolTradesdf){
  #get unique tickers to get data for
  tickers <- unique(anyCapitolTradesdf$asset.assetTicker)
```

```
  #get the earliest and latest date in the dataframe so we can get the entire month of stock returns to
  # Find the earliest and latest dates
  earliest_date <- min(anyCapitolTradesdf$txDate, na.rm = TRUE)
  latest_date <- max(anyCapitolTradesdf$txDate, na.rm = TRUE)

  # Extract the first day of the month of the earliest date
  first_day_of_first_month <- as.Date(format(earliest_date, "%Y-%m-01"))

  # Get the last day of the last month
  last_day_of_last_month <- ceiling_date(latest_date, "month") - days(1)

  from_date <- first_day_of_first_month
  to_date <- last_day_of_last_month

  stock_data <- tq_get(tickers,
                       get  = "stock.prices",
                       from = from_date,
                       to   = to_date  )
  return(stock_data)
}
stockPricesForTickers <- getStockPricesForTickers(scrubbedCapitolTradesData)
```

Finally we validate the capitoltrades dataset by filtering any trades that were not found in our financial stock data collection.

```
removeTradesWithoutStockPrices <- function(anyCapitolTradesdf, anyStockPricesdf){
  #get unique tickers from both datasets
  unique_prices_tickers <- unique(anyStockPricesdf$symbol)
  unique_trades_tickers <- unique(anyCapitolTradesdf$asset.assetTicker)
  #find the difference between the two
  ticker_diff <- setdiff(unique_prices_tickers,unique_trades_tickers)
  #edit first dataset to filter the data
  anyCapitolTradesdf <- anyCapitolTradesdf %>%
    filter(!(asset.assetTicker %in% ticker_diff))
  return(anyCapitolTradesdf)
}

finalScrubbedTradesData <- removeTradesWithoutStockPrices(scrubbedCapitolTradesData, stockPricesForTicke
```

Model Building

This last section will walkthrough the models we built to analyze this data. First we read in the data and modify the date variable to be of "Date" dtype.

```
all_politician_trade_data <- read.csv("Data/Scrubbed_All_CapitolTrades_Data.csv", header = TRUE) %>%
  mutate(txDate = as.Date(txDate))

#filter out any null or "" values of tickers if any
all_politician_trade_data <- all_politician_trade_data[!is.na(all_politician_trade_data$asset.assetTicke
all_politician_trade_data <- all_politician_trade_data[all_politician_trade_data$asset.assetTicker != "
```

```r
all_stock_trade_data <- read.csv("Data/Stock_Prices_For_All_CapitolTrades_Tickers.csv", header = TRUE) %
  mutate(date = as.Date(date))
```

Then we will create several functions which have different purposes:
1) **Calculate_Net_Positions**: Filters the dataframe and groups each ticker together to quantify the net position over our desired timeframe.
2) **Sum_Net_Positions**: Merges dataframes by ticker and replaces null values with zeroes. Then calculates net sum between the two separate positions. Returns a dataframe with just ticker and netposition.
3) **Calculate_Portfolio_Weights_From_Net_Positions**: Filters net positions dataframes and calculates weights for each ticker to use later in the porfolio calculation.

```r
calculate_net_positions <- function(trades_df, start_date = NULL, end_date = NULL) {
  trades_df <- trades_df %>%
    filter(if(!is.null(start_date)) txDate >= start_date else TRUE) %>%
    filter(if(!is.null(end_date)) txDate <= end_date else TRUE)

  net_positions <- trades_df %>%
    group_by(asset.assetTicker) %>%
    summarise(netPosition = sum(ifelse(txType %in% c("buy", "receive"), value, -value)))
    #receive is equal to a buy, exchange is equal to a sell from a portfolio perspective]

  return(net_positions)
}

sum_net_positions <- function(netPosition.x, netPosition.y) {
  # Merge dataframes by assetTicker
  merged_df <- merge(netPosition.x, netPosition.y, by = "asset.assetTicker", all = TRUE, suffixes = c("_

  # Replace NA values with 0
  merged_df[is.na(merged_df)] <- 0

  # Calculate sum of net positions
  merged_df$netPosition <- merged_df$netPosition_1 + merged_df$netPosition_2

  # Remove unnecessary columns
  merged_df <- merged_df[, c("asset.assetTicker", "netPosition")]

  return(merged_df)
}

calculate_portfolio_weights_from_net_positions <- function(net_positions_df) {

  # Calculate net positions
  net_positions <- net_positions_df %>%
    filter(netPosition > 0) %>%
    filter(!is.na(asset.assetTicker) & asset.assetTicker != "")

  # Calculate portfolio weights
  portfolio_weights <- net_positions %>%
    mutate(portfolioWeight = netPosition / sum(netPosition)) %>%
    select(symbol = asset.assetTicker, weight = portfolioWeight)

  return(portfolio_weights)
```

```
}
```

First we create a starting portfolio for the first two quarters of 2021.

```
#create a starting portfolio for the first 2 quarters of 2021
starting_net_positions <- calculate_net_positions(all_politician_trade_data,
                                                  start_date = "2021-03-23",
                                                  end_date = "2021-08-31")
```

We then use the tq package again to calculate monthly returns for each ticker. This may take 15-30 seconds to run.

```
allTickerMonthlyStockReturns <- all_stock_trade_data %>%
  group_by(symbol) %>%
  tq_transmute(adjusted, periodReturn, period = "monthly")
```

Next we define our broad range for the portfolio and create a list of strings for the months within that range. We will use this later for dataframe creation.

```
start_date <- as.Date("2021-09-01")
end_date <- as.Date("2024-02-29")
running_net_positions <- starting_net_positions

monthly_dates <- seq(start_date, end_date, by = "month") %>%
  floor_date(unit = "month") %>%
  unique()
```

We then create an empty dataframe for results to be caught into. And a function for identifying the mode within a column, which we had to use for identifying dates.

```
#Create the empty results dataframe
results <- data.frame(date = as.Date(character()), portfolio_returns = numeric(), stringsAsFactors = FA

#Mode date function for some reason
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}
```

We then use a for loop to analyze each monthly return across the politician data, building a dataframe with each loop.

```
for (i in 1:(length(monthly_dates)-1)) {
  #get the start and end dates for the current month
  month_start <- monthly_dates[i]
  month_end <- monthly_dates[i+1] - days(1)

  #filter the stock trades dataframe for the current month
  month_trades <- all_politician_trade_data %>%
    filter(txDate >= month_start & txDate <= month_end)
```

```
  #update running net position for the current month
  running_net_positions <- sum_net_positions(running_net_positions, calculate_net_positions(month_trades

  #create portfolio weights for the current month
  month_portfolio_weights <- calculate_portfolio_weights_from_net_positions(running_net_positions)

  #create the monthly returns for the current month
  #add tq_get function?
  #pre-create a results table with month_index, and closing dates, use it to handle this months+1
  month_returns <- allTickerMonthlyStockReturns %>%
    filter(date >= month_start & date <= month_end)

  month_portfolio_returns_df <- left_join(month_portfolio_weights, month_returns, by = "symbol") %>%
    mutate(portfolio_return = weight * monthly.returns)

  month_portfolio_return <- month_portfolio_returns_df %>%
    summarise(portfolio_return = sum(portfolio_return, na.rm = TRUE))

  results <- results %>%
    add_row(date = Mode(month_portfolio_returns_df$date[1]), portfolio_returns = month_portfolio_return$
}
```

We use the tq package again to gather general stock market results from the S&P 500 to compare our porfolio's return against. Merge the two dataframes by date to then linear regression.

```
snp_market_results <- tq_get("^GSPC", get="stock.prices", from = start_date , to = end_date) %>%
  tq_transmute(adjusted, periodReturn, period = "monthly")

#append market results to
results <- merge(results, snp_market_results, by="date")

results <- results %>%
  rename(snp_market_returns = monthly.returns)
```

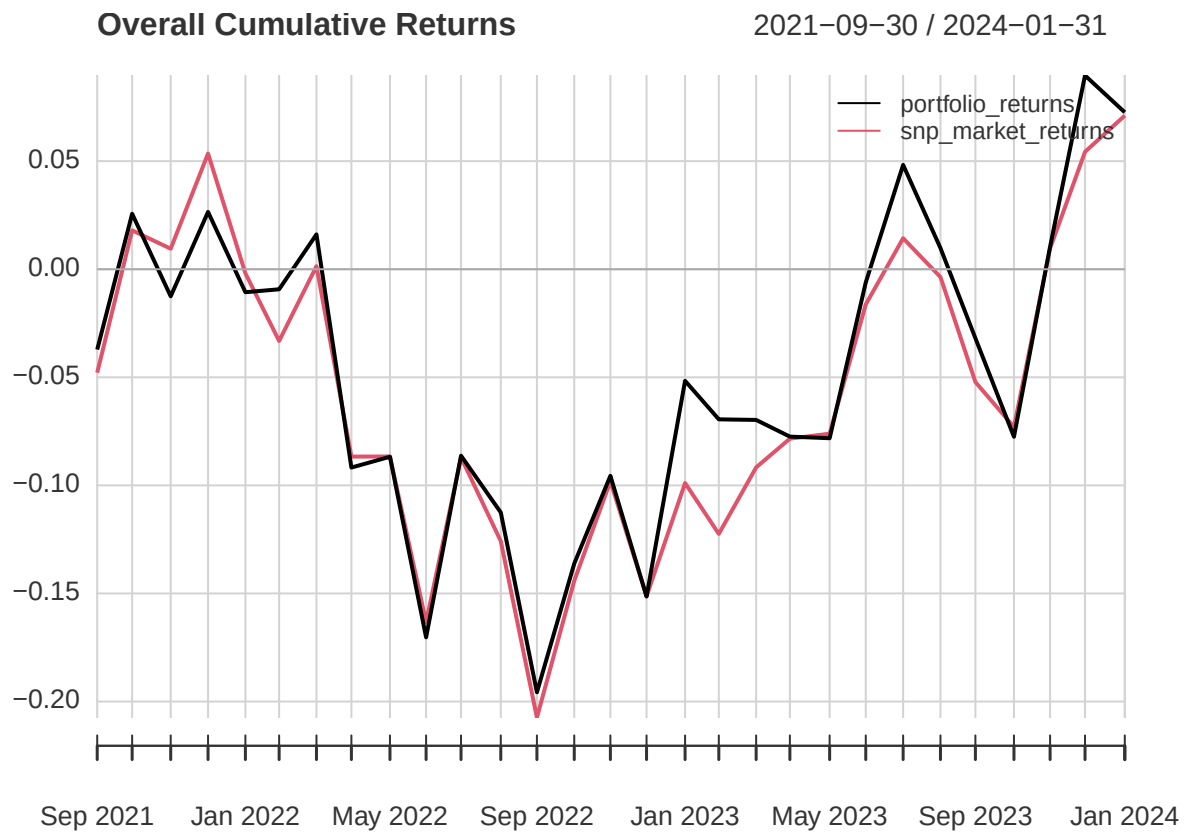
Finally, we have our linear regression model which shows the Beta and R^2, examining how our politician's portfolio's follow the market and/or beat it.

```
beta_model <- lm(portfolio_returns ~ snp_market_returns, data=results)
summary(beta_model)
```

```
##
## Call:
## lm(formula = portfolio_returns ~ snp_market_returns, data = results)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.037322 -0.012161  0.000032  0.009978  0.052496
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.0001972  0.0039146    0.05     0.96
## snp_market_returns 1.0509811  0.0723671   14.52 2.81e-14 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02103 on 27 degrees of freedom
## Multiple R-squared:  0.8865, Adjusted R-squared:  0.8823
## F-statistic: 210.9 on 1 and 27 DF,  p-value: 2.807e-14
```

```r
chart.CumReturns(results,
                 geometric = TRUE,
                 legend.loc = "topright",
                 #plot.engine = "plotly",
                 main = "Overall Cumulative Returns")
```




```r
#convert to timeseries from dataset
results_portfolio_returns_xts <- xts(x = results$portfolio_returns, order.by = results$date)
results_snp_market_returns_xts <- xts(x = results$snp_market_returns, order.by = results$date)

#get cumulative returns
portfolio_cum_returns <- Return.cumulative(results_portfolio_returns_xts, geometric = TRUE)
paste("Cumulative returns for overall portfolio is:", round(portfolio_cum_returns*100,2),"%")
```

```
## [1] "Cumulative returns for overall portfolio is: 7.26 %"
```

```
snp_market_cum_returns <- Return.cumulative(results_snp_market_returns_xts, geometric = TRUE)
paste("Cumulative returns for S&P market is:", round(snp_market_cum_returns*100,2),"%")
```

```
## [1] "Cumulative returns for S&P market is: 7.11 %"
```

Additional Filtering

Age Filtering

To provide some extra insight, we also performed similar portfolio analysis based on age within our data. We separated politicians into three age groups: 40-60, 60-70 and 70+. The analysis is pretty much the same as the first model.

This first code block removes null date of birth values (dob), calculates ages for each of the politicians and groups them into the age groups specified.

```
clean_age_politician_data = all_politician_trade_data

clean_age_politician_data$Age = as.Date(clean_age_politician_data$politician.dob) #Convert to age for n

clean_age_politician_data$Age = as.integer(age_calc(clean_age_politician_data$Age, units = "years")) #C

age_df = clean_age_politician_data %>% mutate(age_group = case_when(Age >= 40 & Age < 60 ~ "40_60_Years
                                                                    Age >= 60 & Age < 70 ~ "60_70_Years
                                                                  TRUE ~ "70+_Years_Old"))
```

We then create a variable to capture the unique age_groups and initialize an empty dataframe just as we did before.

```
age_groups = unique(age_df$age_group)
portfolio = data.frame(date = as.Date(character()), portfolio_returns = numeric(), stringsAsFactors = F
```

Now we run through each age group, filter the dataframe to only contain those age groups, and perform the portfolio analysis as we did before.

```
for (grouping in age_groups) {
  ages = grouping
  # print(ages)
  filtered_df <- age_df %>%
    filter(age_group == ages)
  starting_net_positions <- calculate_net_positions(filtered_df, start_date = "2021-03-23", end_date = "
  #Specify the date range for the analysis
  start_date <- as.Date("2021-09-01")
  end_date <- as.Date("2024-02-29")
  running_net_positions <- starting_net_positions
  monthly_dates <- seq(start_date, end_date, by = "month") %>%
    floor_date(unit = "month") %>%
    unique()
  #Create the empty results dataframe
  results <- data.frame(date = as.Date(character()), portfolio_returns = numeric(), stringsAsFactors = F

  for (i in 1:(length(monthly_dates)-1)) {
```

```
    #get the start and end dates for the current month
    month_start <- monthly_dates[i]
    month_end <- monthly_dates[i+1] - days(1)

    #filter the stock trades dataframe for the current month
    month_trades <- filtered_df %>%
      filter(txDate >= month_start & txDate <= month_end)

    #update running net position for the current month
    running_net_positions <- sum_net_positions(running_net_positions, calculate_net_positions(month_trac

    #create portfolio weights for the current month
    month_portfolio_weights <- calculate_portfolio_weights_from_net_positions(running_net_positions)

    #create the monthly returns for the current month
    #add tq_get function?
    #pre-create a results table with month_index, and closing dates, use it to handle this months+1
    month_returns <- allTickerMonthlyStockReturns %>%
      filter(date >= month_start & date <= month_end)

    month_portfolio_returns_df <- left_join(month_portfolio_weights, month_returns, by = "symbol") %>%
      mutate(portfolio_return = weight * monthly.returns)

    month_portfolio_return <- month_portfolio_returns_df %>%
      summarise(portfolio_return = sum(portfolio_return, na.rm = TRUE))

    results <- results %>%
      add_row(date = Mode(month_portfolio_returns_df$date), portfolio_returns = month_portfolio_return$p
  }
  if(ages == "40_60_Years_Old"){
    portfolio = results
  }else{
    portfolio = merge(portfolio, results, by = "date")
  }
}

names(portfolio)[2:4] = age_groups #Rename columns
```

Merge the dataframes of our “Market Portfolio” and the age group portfolio data.

```
snp_market_results <- tq_get("^GSPC", get="stock.prices", from = start_date , to = end_date) %>%
  tq_transmute(adjusted, periodReturn, period = "monthly")

#append market results to
all_ages_portfolio <- merge(portfolio, snp_market_results, by="date")

all_ages_portfolio <- all_ages_portfolio %>%
  rename(snp_market_returns = monthly.returns)
```

Create linear regression models to characterize the beta for each age group.

```r
model_40_60 = lm(all_ages_portfolio$"40_60_Years_Old" ~ snp_market_returns, data = all_ages_portfolio)
model_60_70 = lm(all_ages_portfolio$"60_70_Years_Old" ~ snp_market_returns, data = all_ages_portfolio)
model_70 = lm(all_ages_portfolio$"70+_Years_Old" ~ snp_market_returns, data = all_ages_portfolio)

beta_40_60 = summary(model_40_60)$coefficients[2, 1]
beta_60_70 = summary(model_60_70)$coefficients[2, 1]
beta_70 = summary(model_70)$coefficients[2, 1]

Betas = data.frame("Age Group" = age_groups, Betas = c(beta_40_60, beta_60_70, beta_70))

Betas #Betas for each age group
```

```
##         Age.Group     Betas
## 1 40_60_Years_Old 0.9973172
## 2 60_70_Years_Old 1.0134017
## 3   70+_Years_Old 1.3080336
```

```r
portfolio_40_60_returns_xts <- xts(x = all_ages_portfolio$"40_60_Years_Old", order.by = all_ages_portfol
portfolio_60_70_market_returns_xts <- xts(x = all_ages_portfolio$"60_70_Years_Old", order.by = all_ages_
portfolio_70_returns_xts <- xts(x = all_ages_portfolio$"70+_Years_Old", order.by = all_ages_portfolio$da
snp_returns_xts <- xts(x = all_ages_portfolio$snp_market_returns, order.by = all_ages_portfolio$date)

#get cumulative returns
portfolio_40_60_cum_returns <- Return.cumulative(portfolio_40_60_returns_xts, geometric = TRUE)
paste("Cumulative returns for Politicians Age 40-60 portfolio is:", round(portfolio_40_60_cum_returns*10
```

```
## [1] "Cumulative returns for Politicians Age 40-60 portfolio is: 11.18 %"
```

```r
portfolio_60_70_cum_returns <- Return.cumulative(portfolio_60_70_market_returns_xts, geometric = TRUE)
paste("Cumulative returns for Politicians Age 60-70 portfolio is:", round(portfolio_60_70_cum_returns*10
```

```
## [1] "Cumulative returns for Politicians Age 60-70 portfolio is: 12.43 %"
```

```r
portfolio_70_cum_returns <- Return.cumulative(portfolio_70_returns_xts, geometric = TRUE)
paste("Cumulative returns for Politicians Age 70+ portfolio is:", round(portfolio_70_cum_returns*100,2)
```

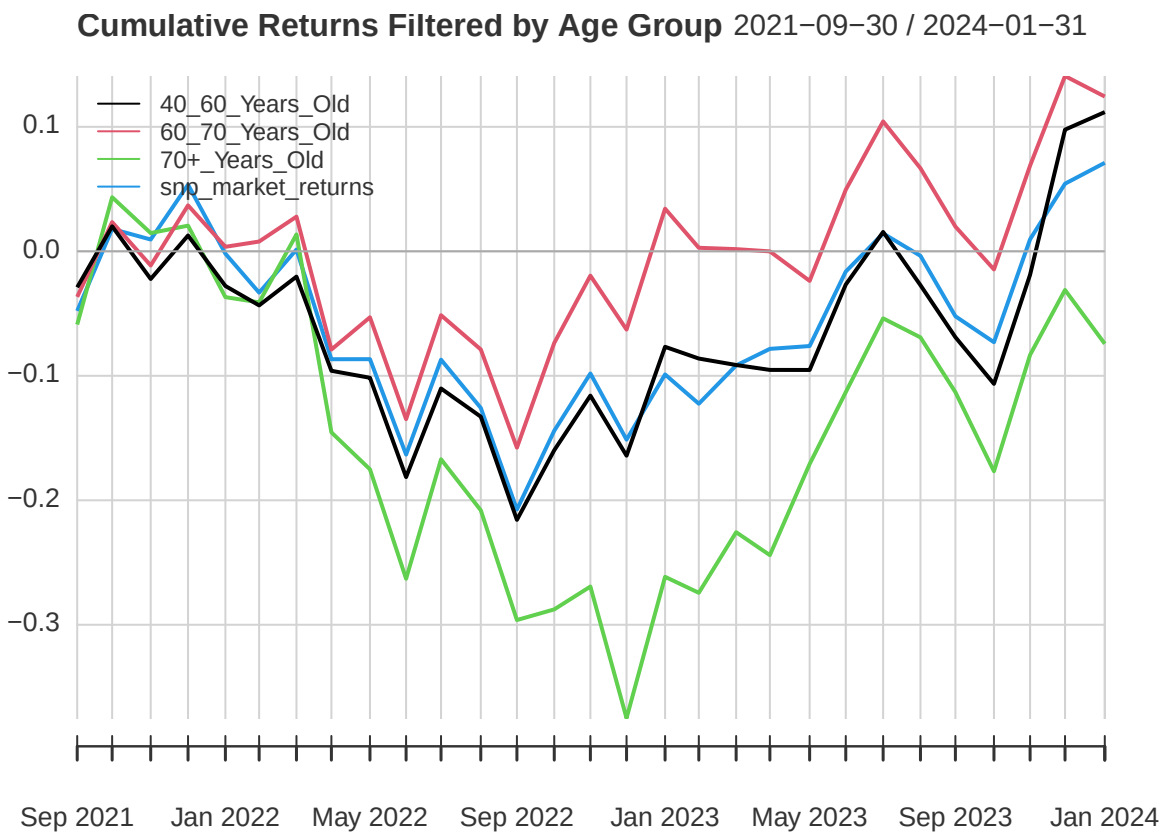
```
## [1] "Cumulative returns for Politicians Age 70+ portfolio is: -7.43 %"
```

```r
snp_market_cum_returns <- Return.cumulative(snp_returns_xts, geometric = TRUE)
paste("Cumulative returns for S&P market is:", round(snp_market_cum_returns*100,2),"%")
```

```
## [1] "Cumulative returns for S&P market is: 7.11 %"
```

Visualize cumulative returns.

```r
chart.CumReturns(all_ages_portfolio, geometric = TRUE, legend.loc = "topleft", main = "Cumulative Retur
```



Additional Filtering ### Party Filtering Now we will walk through the filtering by party analysis that was included in the final report. First we create a column for the party of each individual in our politician trade dataframe.

```
party_df = all_politician_trade_data %>%
  mutate(party_group = case_when(politician.party == 'republican' ~ "Republican",
                                 politician.party == 'democrat' ~ "Democrat",
                                 TRUE ~ "Third Party/Other"))

party_groups = unique(party_df$party_group) #Gather party names
portfolio = data.frame(date = as.Date(character()), portfolio_returns = numeric(), stringsAsFactors = F
```

Next, following the same principles from the filter by age analysis, we collect monthly data on the portfolio of politicians based on their party.

```
for (grouping in party_groups) {
  parties = grouping
  filtered_df <- party_df %>%
    filter(party_group == parties)
  starting_net_positions <- calculate_net_positions(filtered_df, start_date = "2021-03-23", end_date =
  #Specify the date range for the analysis
  start_date <- as.Date("2021-09-01")
  end_date <- as.Date("2024-02-29")
  running_net_positions <- starting_net_positions
  monthly_dates <- seq(start_date, end_date, by = "month") %>%
    floor_date(unit = "month") %>%
```

```
    unique()
  #Create the empty results dataframe
  results <- data.frame(date = as.Date(character()), portfolio_returns = numeric(), stringsAsFactors = F

  for (i in 1:(length(monthly_dates)-1)) {
    #get the start and end dates for the current month
    month_start <- monthly_dates[i]
    month_end <- monthly_dates[i+1] - days(1)

    #filter the stock trades dataframe for the current month
    month_trades <- filtered_df %>%
      filter(txDate >= month_start & txDate <= month_end)

    #update running net position for the current month
    running_net_positions <- sum_net_positions(running_net_positions, calculate_net_positions(month_trad

    #create portfolio weights for the current month
    month_portfolio_weights <- calculate_portfolio_weights_from_net_positions(running_net_positions)

    #create the monthly returns for the current month
    #add tq_get function?
    #pre-create a results table with month_index, and closing dates, use it to handle this months+1
    month_returns <- allTickerMonthlyStockReturns %>%
      filter(date >= month_start & date <= month_end)

    month_portfolio_returns_df <- left_join(month_portfolio_weights, month_returns, by = "symbol") %>%
      mutate(portfolio_return = weight * monthly.returns)

    month_portfolio_return <- month_portfolio_returns_df %>%
      summarise(portfolio_return = sum(portfolio_return, na.rm = TRUE))

    results <- results %>%
      add_row(date = Mode(month_portfolio_returns_df$date), portfolio_returns = month_portfolio_return$p
  }
  if(parties == "Republican"){
    portfolio = results
  }else{
    portfolio = merge(portfolio, results, by = "date")
  }
}
```

Again we collect data on the general S&P 500 for a comparison.

```
names(portfolio)[2:4] = party_groups #Rename columns


#get the market results
snp_market_results <- tq_get("^GSPC", get="stock.prices", from = start_date , to = end_date) %>%
  tq_transmute(adjusted, periodReturn, period = "monthly")

#append market results to
all_parties_portfolio <- merge(portfolio, snp_market_results, by="date")
```

```r
all_parties_portfolio <- all_parties_portfolio %>%
  rename(snp_market_returns = monthly.returns)
```

We find the beta for each group through linear regression.

```r
model_repubublican = lm(all_parties_portfolio$"Republican" ~ snp_market_returns, data = all_parties_por
model_democrat = lm(all_parties_portfolio$"Democrat" ~ snp_market_returns, data = all_parties_portfolio
model_other = lm(all_parties_portfolio$"Third Party/Other" ~ snp_market_returns, data = all_parties_por

beta_republican = summary(model_repubublican)$coefficients[2, 1]
beta_democrat = summary(model_democrat)$coefficients[2, 1]
beta_other = summary(model_other)$coefficients[2, 1]

Betas = data.frame("Parties" = c("beta_republican", "beta_democrat", "beta_other"), Betas = c(beta_repu

Betas
```

```
##           Parties     Betas
## 1 beta_republican 0.9694591
## 2   beta_democrat 1.2172948
## 3      beta_other 1.2191616
```

```r
republican_portfolio_returns_xts <- xts(x = all_parties_portfolio$"Republican", order.by = all_parties_
democrat_snp_market_returns_xts <- xts(x = all_parties_portfolio$"Democrat", order.by = all_parties_por
other_snp_market_returns_xts <- xts(x = all_parties_portfolio$"Third Party/Other", order.by = all_parti
snp_returns_xts <- xts(x = all_parties_portfolio$snp_market_returns, order.by = all_parties_portfolio$da

#get cumulative returns
republican_cum_returns <- Return.cumulative(republican_portfolio_returns_xts, geometric = TRUE)
paste("Cumulative returns for Republicans portfolio is:", round(republican_cum_returns*100,2),"%")
```

```
## [1] "Cumulative returns for Republicans portfolio is: 14.23 %"
```

```r
democrat_cum_returns <- Return.cumulative(democrat_snp_market_returns_xts, geometric = TRUE)
paste("Cumulative returns for Democrats portfolio is:", round(democrat_cum_returns*100,2),"%")
```

```
## [1] "Cumulative returns for Democrats portfolio is: 2.82 %"
```

```r
other_portfolio_cum_returns <- Return.cumulative(other_snp_market_returns_xts, geometric = TRUE)
paste("Cumulative returns for 'Others' portfolio is:", round(other_portfolio_cum_returns*100,2),"%")
```

```
## [1] "Cumulative returns for 'Others' portfolio is: -25.21 %"
```

```r
snp_market_cum_returns <- Return.cumulative(snp_returns_xts, geometric = TRUE)
paste("Cumulative returns for S&P market is:", round(snp_market_cum_returns*100,2),"%")
```

```
## [1] "Cumulative returns for S&P market is: 7.11 %"
```

And finally we visualize the cumulative returns.

```
chart.CumReturns(all_parties_portfolio, geometric = TRUE, legend.loc = "topleft", main = "Cumulative Re
```