Mean-Variance Portfolio Optimization Report

Group T1: Lauren Newton, Truc Tran, Chavi Pathak,

Joshua Katana, Francisco Fuentes

ISDS 570

12/15/2024

Table of Contents

Executive Summary

This project demonstrates the successful implementation of Mean-Variance (MV) portfolio optimization to achieve an ideal balance between risk and return. By leveraging historical stock data from 2016 to 2020 and applying Markowitz's MV framework, our team optimized a 15-stock portfolio with significant allocations to WM, KMB, and ZNGA while taking short positions in CSOD, FBC, and FELE to hedge against market risks. The optimized portfolio significantly outperformed the S&P 500 Total Return Index (SP500TR) during the testing phase in the first quarter of 2021, delivering higher annualized returns, lower volatility, and nearly double the Sharpe ratio compared to the benchmark. These results highlight the effectiveness of the optimization strategy in delivering superior risk-adjusted returns while adapting to market fluctuations. This report outlines the methodology, data processing steps, and analytical techniques employed to achieve these outcomes, offering valuable insights for future portfolio management strategies.

I. Introduction

The principles of Mean-Variance (MV) optimization form the foundation of modern portfolio theory, enabling investors to achieve a balance between maximizing returns and managing risks. This report explores the application of MV optimization to construct and test an investment portfolio comprising 15 stocks across diverse industries. Historical data from 2016 to 2020 was analyzed to develop an optimized portfolio that aligns with investor objectives while adhering to constraints such as full investment of capital and the incorporation of short selling.

The portfolio's performance was benchmarked against the S&P 500 Total Return Index (SP500TR) during the testing period in the first quarter of 2021, revealing its ability to outperform the benchmark in terms of annualized returns, volatility, and risk-adjusted metrics such as the Sharpe ratio. This report provides a detailed discussion of the data sources, preprocessing steps, portfolio optimization techniques, and performance evaluation metrics used throughout the project. The findings underscore the potential of data-driven methodologies in creating resilient and high-performing investment strategies, even in volatile market conditions.

STOCK MARKET TICKERS

Serial No.	Ticker	Description/ Company Name
1	CSOD	Cornerstone OnDemand
2	FAST	Fastenal Company
3	FBC	Flagstar Bancorp, Inc.
4	FELE	Franklin Electric Co., Inc.
5	GGB	Gerdau S.A.
6	HD	The Home Depot, Inc.
7	IDCC	InterDigital, Inc.
8	IRM	Iron Mountain Incorporated
9	KMB	Kimberly-Clark Corporation
10	MLM	Martin Marietta Materials, Inc.
11	NP	Neenah, Inc.
12	PZN	Pzena Investment Management, Inc.
13	SLF	Sun Life Financial, Inc.
14	WM	Waste Management, Inc.
15	ZNGA	Zynga Inc.

II. ETL Process

a. Data Sources (Excel/Internet sources)

Our project relies on two primary data sources: individual stock data and a benchmark index. The benchmark index data was obtained from the Yahoo Finance SP500TR Historical Data website, comprising of daily adjusted closing quotes for the period starting on January 1st 2016 and ending on March 26th 2021. This dataset also includes open, high, low and close prices, as well as trading volume (set to 0), for the stock ticker/symbol SP500TR. The individual stock data comes from the eod.csv source provided by the professor which is comprised of daily adjusted quotes for the period starting on January 1st 2016 and ending on March 26th 2021. This dataset also includes open, high, low, and close prices, as well as adjusted volume, for several stock tickers. The data serves as the foundation for calculating portfolio returns and conducting comparative analysis.

Additionally, a custom trading calendar was utilized, derived from an Excel sheet provided in the course materials. The calendar was modified to include trading days and holidays during the project period, along with additional columns identifying the last trading day of the month and the previous trading day. This calendar ensures alignment of stock and benchmark data with trading dates, facilitating accurate time-series analysis.

b. Database Setup (in PostgreSQL)

A PostgreSQL database, named *stockmarket*, was created to store, organize, and query stock data, benchmark index data, and the custom trading calendar. This centralized database ensures efficient data management and accessibility for analysis.

The first step in the setup involved creating the table, *eod_quotes*, to store stock data. This table includes columns for the stock ticker, date, adjusted open, high, low, close prices, and adjusted trading volume. The combination of ticker and date serves as the primary key. Approximately 17 million rows of stock data were imported into this table from a CSV file.

Similarly, a table named *eod_indices* was created to store benchmark index data for the SP500TR. The table contains columns for the index symbol, date, open, high, low, close, adjust close prices, and volume, with a primary key formed by the combination of symbol and date. Data for this table was collected from Yahoo Finance, formatted in Excel, and imported into PostgreSQL.

A third table, *custom_calendar*, was created to manage the trading calendar. This table includes columns for the date, year, month, day, and day of the week, as well as flags for trading days, end-of-month days, and the previous trading day. The calendar was initially prepared in Excel, with non-trading days and holidays added manually for the project period. After importing the data into PostgreSQL, additional columns were populated using SQL queries to identify end-of-month days and the previous trading day for each date.

To facilitate efficient querying and ensure data security during analysis in R, a read-only user role, *stockmarketreader*, was created within the PostgreSQL database. This role was assigned SELECT privileges for all existing tables and views in the public schema, along with default privileges for any future tables. By using this role, the R script used later can securely connect to the stockmarket database, execute SQL queries to retrieve necessary data, and maintain database integrity without the risk of unintentional modifications.

c. **Data Cleaning and Preprocessing (in R)**

The data cleaning and preprocessing for the stock market case in R involved several key steps to ensure the dataset was prepared for analysis. Initially, a connection was established to the PostgreSQL database using the RPostgres package. Data was queried from two key tables, *eod_indices* and *eod_quotes*, covering the period between December 31st, 2015, and March 26th, 2021 and restricting tickers to those assigned to all group members in Homework #2. To align with the analysis requirements, additional adjustments, such as inserting missing data for specific dates, were made directly in the dataset.

A custom calendar of trading days was integrated to filter the data and ensure alignment with market trading days. Completeness of the dataset was assessed by calculating the percentage of available data for each symbol relative to the total number of trading days. Only symbols with a data completeness above a set threshold of 0.99 were selected for further analysis.

The reshape2 package was used to transform the dataset into a pivot table format, with dates as rows and symbols as columns. Missing data was handled through forward-filling imputation using the zoo package's na.locf function, limiting gaps to a maximum of three days. After imputation, daily returns were calculated for each stock using the Performance Analytics package, ensuring consistent data formatting for subsequent analysis. Symbols with extreme returns (daily returns exceeding 100%) were excluded to maintain data reliability. The resulting data was transformed into an xts format for time-series analysis, forming the basis for the portfolio optimization.

III. **Portfolio Optimization**

Portfolio optimization employed Markowitz's Mean-Variance (MV) framework to determine the optimal allocation of assets. The analysis was divided into training and testing phases. The training set included all data up to the last 58 trading days, with the testing set comprising the remaining 58 days. For the training phase, the annualized return for the SP500TR Index (our benchmark) was calculated, and its mean return was set as the minimum acceptable return (MAR).

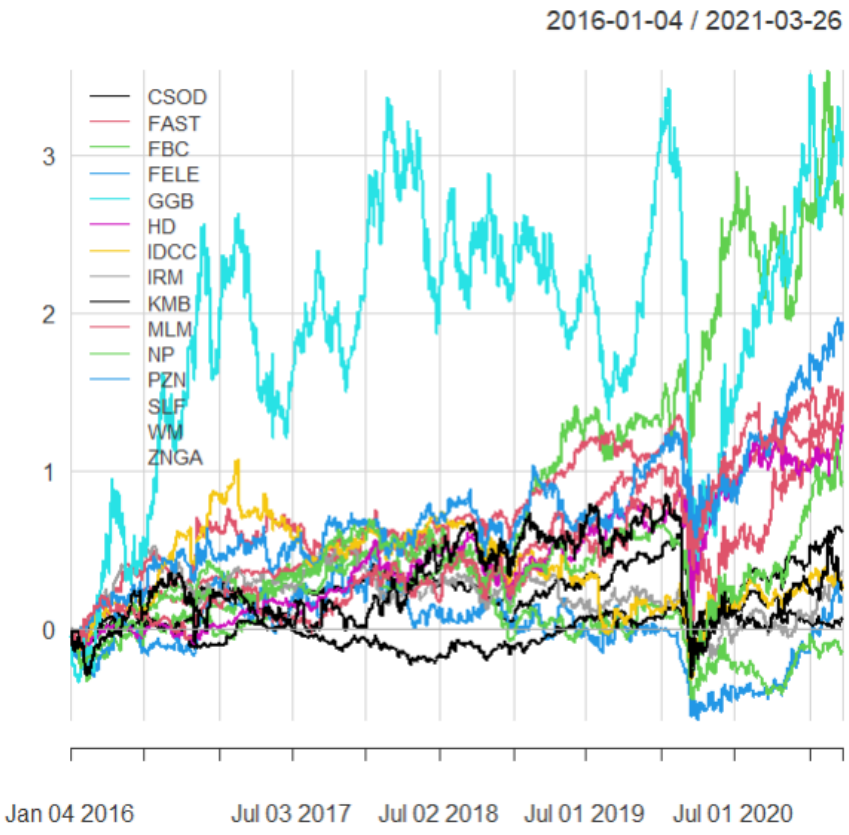
Using the PortfolioAnalytics package, a portfolio specification was defined, incorporating objectives such as minimizing risk (measured by standard deviation) and achieving returns that met or exceeded the MAR. Constraints included full investment of capital and ensuring portfolio aligned with the predefined objectives. The ROI optimization method was used to solve the portfolio allocation problem, generating weights for each asset. These weights, including allowances for short selling, were applied to the testing data to calculate the hypothetical portfolio's returns.

Performance evaluation compared the portfolio's cumulative and annualized returns to the benchmark, revealing insights into its effectiveness. The visualization of cumulative returns was achieved using the chart.CumReturns function, highlighting the portfolio's growth over time and facilitating a direct comparison with the benchmark.

IV. Results and Analysis

a. Cumulative Return for Project Range #1

The chart below illustrates the cumulative returns for the selected stocks between 2016 and 2020.



Based on the cumulative returns chart, we can observe a range of performances across the stocks. Among the highest performers, GGB stands out with a remarkable cumulative return of 3.15, signaling exceptional growth over the period. Similarly, ZNGA achieved a cumulative return of approximately 2.75, showcasing robust growth in their sector. Other notable stocks include FELE, which recorded a return of approximately 1.95, and MLM with a return of approximately 1.59, both reflecting strong performance in their respective sectors.

Stocks like FAST and WM also displayed solid returns, reflecting consistent performance in their industries. HD showed moderate growth with a cumulative return of approximately 1.30, indicative of steady performance within the home improvement sector. On the other hand, IRM and SLF saw more subdued returns of approximately 0.38

and 0.63, respectively. Although positive, these returns are lower compared to the standout performers in the group.

The lowest cumulative returns were observed in KMB with a return of approximately 0.08, and NP with -0.13, indicating some underperformance in their respective sectors during this period.

b. **Optimized Portfolio Weights and Sums for Project Range #1**

The table below displays the optimized portfolio weights for each stock, based on Project Range #1 (2016-2020). The sum of these weights is 1.0000, confirming that the portfolio is fully allocated across the selected assets. Negative weights indicate short positions, while positive weights represent long positions.

Symbol	Weight
CSOD	-0.01498
FAST	0.07056
FBC	-0.02029
FELE	-0.03902
GGB	-0.00899
HD	0.06737
IDCC	-0.01203
IRM	0.00776
KMB	0.22330
MLM	0.03639
NP	0.03409
PZN	-0.01324
SLF	0.06652
WM	0.43957
ZNGA	0.16299
Sum	**1.00000**

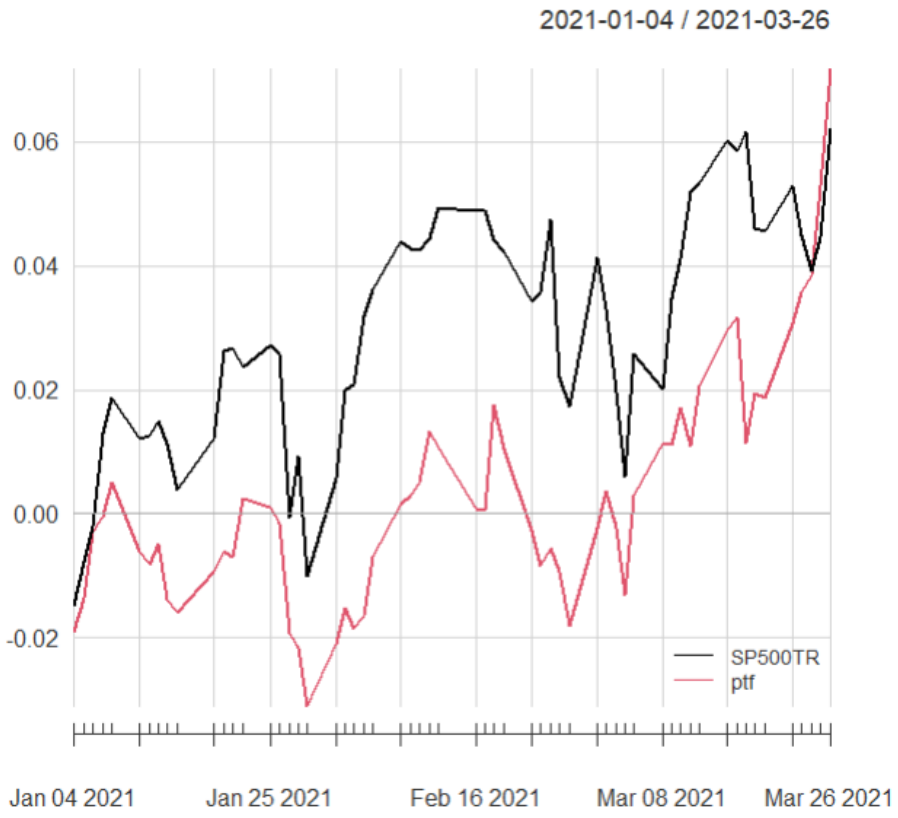
Optimized Portfolio Weight Distribution for Project Range #1 (2016-2020)

Positive weights indicate long positions, where the portfolio benefits from price increases in the respective stocks, while negative weights represent short positions, profiting from price declines. The distribution reflects the optimization process aimed at maximizing returns while minimizing risk over the specified period.

The portfolio heavily emphasizes WM, with a weight of approximately 0.44, indicating its significant contribution to the portfolio's overall performance. Stocks such as KMB and ZNGA also hold notable weights of approximately 0.22 and 0.16, respectively. Conversely, several stocks, including CSOD, FBC, and FELE, are allocated negative weights, suggesting short positions in these assets to optimize the portfolio's returns.

c. **Portfolio Performance for Project Range #2**

The cumulative return chart compares the performance of the optimized portfolio against the SP500TR index for all available 2021 data. The black line represents the SP500TR index, which initially outperforms the optimized portfolio throughout most of the year. However, in the last days of the 2021 testing period, the portfolio surpasses the SP500TR index, indicating strong performance in the latter part of the quarter.



d. **Annualized Portfolio and SP500TR Index Returns for Project Range #2**

The table compares key performance metrics between the S&P 500 Total Return (SP500TR) index and the optimized portfolio (ptf). Metrics include annualized return, annualized standard deviation (risk), and the annualized Sharpe ratio (assuming a risk-free rate of 0%). The results highlight the portfolio's superior risk-adjusted performance compared to the benchmark.

	SP500TR	ptf
Annualized Retturn	0.29870	0.35140
Annualized Std Dev	0.16230	0.14180
Annualized Sharpe (Rf=0%)	1.83990	2.47910

Performance Metrics Comparison: S&P 500 Total Return vs. Portfolio

The optimized portfolio achieved a higher annualized return compared to the SP500TR index, suggesting that the allocation strategy was effective in delivering higher growth. The portfolio exhibited lower volatility than the SP500TR index, indicating a more stable performance despite delivering higher returns. This is a favorable attribute for risk-averse investors. Lastly, the optimized portfolio achieved a Sharpe ratio nearly double that of the SP500TR index, reflecting its superior risk-adjusted return.

V. Conclusion

Our project successfully utilized Mean-Variance (MV) principles to optimize a 15-stock investment portfolio, spanning various industries, for the period of 2016 to 2020 and tested its performance against the benchmark S&P 500 Total Return Index (SP500TR) using data from the first quarter of 2021. The optimized portfolio exhibited strong cumulative and annualized returns, lower volatility, and a significantly higher Sharpe ratio compared to the SP500TR index, indicating its superior risk-adjusted performance. The portfolio heavily emphasized WM(44%), KMB(22%), and ZNGA(16%), with significant weights, while also including short positions in assets like CSOD, FBC, and FELE to optimize returns. Our optimized portfolio successfully achieved a higher annualized return compared to the SP500TR index, explaining that the allocation strategy was effective in delivering higher growth. Additionally, this portfolio exhibited lower volatility than the SP500TR index, indicating a more stable performance despite delivering higher returns. The results highlighted the effectiveness of the allocation strategy in delivering higher growth, stability, and superior risk-adjusted performance, making it favorable for risk-averse investors.

Appendices

SQL Code

```
-----------------------------------------------

-- Step 1: Create a database "stockmarket" -----

-----------------------------------------------

-----------------------------------------------

-- Step 2: Import EOD (End of Day) Quotes ------

-----------------------------------------------

-- Create table eod_quotes

/* -- LIFELINE -- DROP TABLE public.eod_quotes;

CREATE TABLE public.eod_quotes ( ticker character varying(16) COLLATE
pg_catalog."default" NOT NULL, date date NOT NULL, adj_open real,
adj_high real, adj_low real, adj_close real, adj_volume numeric,
CONSTRAINT eod_quotes_pkey PRIMARY KEY (ticker, date) ) WITH ( OIDS =
FALSE ) TABLESPACE pg_default;

ALTER TABLE public.eod_quotes OWNER to postgres; */

-- Import eod.csv to the table

-- Checks

SELECT * FROM eod_quotes LIMIT 10; SELECT COUNT(*) FROM eod_quotes;


-------------------------------------------------------------------

-- Step 3: Import 2016-2021(March) SP500TR data from Yahoo ---------

-------------------------------------------------------------------

-- Navigate to Yahoo's historical data website:
https://finance.yahoo.com/quote/%5ESP500TR/history?p=^SP500TR

-- Set dates to 2016-01-01 and 2021-03-26, copy and paste data to an
excel

-- Save that excel as "SP500TR_PROJECT"

-- Create table to store data and then import it

/*

LIFELINE:

-- DROP TABLE public.eod_indices;
```

```
CREATE TABLE public.eod_indices ( symbol character varying(16) COLLATE
pg_catalog."default" NOT NULL, date date NOT NULL, open real, high
real, low real, close real, adj_close real, volume double precision,
CONSTRAINT eod_indices_pkey PRIMARY KEY (symbol, date) ) WITH ( OIDS =
FALSE ) TABLESPACE pg_default;

ALTER TABLE public.eod_indices OWNER to postgres;

*/

-- Check

SELECT * FROM eod_indices LIMIT 10;

---------------------------------------------------------------

-- Step 4: Prepare a custom calendar (using Excel) ---------

---------------------------------------------------------------

-- Use the professor's custom calendar excel sheet available

-- Modify this excel to start on 2016-01-01 and end on 2021-03-26

-- Modify excel so that the three market holidays of New Year's Day
(1/1/21), MLK Jr Day (1/18/21), and President's Day (2/15/21) are
captured

-- Import this modified custom_calendar.csv to a new table

/* LIFELINE:

-- DROP TABLE public.custom_calendar;

CREATE TABLE public.custom_calendar ( date date NOT NULL, y integer, m
integer, d integer, dow character varying(3) COLLATE
pg_catalog."default", trading smallint, CONSTRAINT
custom_calendar_pkey PRIMARY KEY (date) ) WITH ( OIDS = FALSE )
TABLESPACE pg_default;

ALTER TABLE public.custom_calendar OWNER to postgres;

*/

-- CHECK:

SELECT * FROM custom_calendar LIMIT 10;

--Add columns for later returns analysis: eom (end-of-month) and
prev_trading_day

/*

-- LIFELINE ALTER TABLE public.custom_calendar ADD COLUMN eom
smallint;
```

```
ALTER TABLE public.custom_calendar ADD COLUMN prev_trading_day date;
*/

-- CHECK:

SELECT * FROM custom_calendar LIMIT 10;

-- After creating table - import csv file into table

--Identify trading days

SELECT * FROM custom_calendar WHERE trading=1;

-- Identify previous trading days via a nested query

SELECT date, (SELECT MAX(CC.date) FROM custom_calendar CC WHERE
CC.trading=1 AND CC.date<custom_calendar.date) ptd FROM
custom_calendar ORDER BY date;

-- Update the table with new data UPDATE custom_calendar

SET prev_trading_day = PTD.ptd FROM (SELECT date, (SELECT MAX(CC.date)
FROM custom_calendar CC WHERE CC.trading=1 AND
CC.date<custom_calendar.date) ptd FROM custom_calendar) PTD WHERE
custom_calendar.date = PTD.date;

-- CHECK

SELECT * FROM custom_calendar ORDER BY date;

-- Add the last trading day of 2015 (as the end of the month)

INSERT INTO custom_calendar VALUES('2015-12-
31',2015,12,31,'Thu',1,1,NULL);

-- Re-run the update

-- CHECK again

SELECT * FROM custom_calendar ORDER BY date;

-- Identify the end of the month

SELECT CC.date,CASE WHEN EOM.y IS NULL THEN 0 ELSE 1 END endofm FROM
custom_calendar CC LEFT JOIN (SELECT y,m,MAX(d) lastd FROM
custom_calendar WHERE trading=1 GROUP by y,m) EOM ON CC.y=EOM.y AND
CC.m=EOM.m AND CC.d=EOM.lastd;

-- Update the table with new data

UPDATE custom_calendar SET eom = EOMI.endofm FROM (SELECT CC.date,CASE
WHEN EOM.y IS NULL THEN 0 ELSE 1 END endofm FROM custom_calendar CC
LEFT JOIN (SELECT y,m,MAX(d) lastd FROM custom_calendar WHERE
trading=1 GROUP by y,m) EOM ON CC.y=EOM.y AND CC.m=EOM.m AND
CC.d=EOM.lastd) EOMI WHERE custom_calendar.date = EOMI.date;
```

```
-- CHECK

SELECT * FROM custom_calendar ORDER BY date; SELECT * FROM
custom_calendar WHERE eom=1 ORDER BY date;

--------------------------------------------

-- Step 5: Prepare roles for database -------

--------------------------------------------

-- rolename: stockmarketreader

-- password: read123

/* -- LIFELINE:

-- REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM
stockmarketreader;

-- DROP USER stockmarketreader;

CREATE USER stockmarketreader WITH LOGIN NOSUPERUSER NOCREATEDB
NOCREATEROLE INHERIT NOREPLICATION CONNECTION LIMIT -1 PASSWORD
'read123'; */

-- Grant read rights (on existing tables and views)

GRANT SELECT ON ALL TABLES IN SCHEMA public TO stockmarketreader;

-- Grant read rights (for future tables and views)

ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO
stockmarketreader;

---------------------------------

-- End of Database Portion -------

---------------------------------
```

R Code

```
#rm(list=ls(all=T)) this removes everything from memory if necessary
#step 1: connect to PostgreSQL---------------------------------------

require(RPostgres)
require(DBI)
conn <- dbConnect(RPostgres::Postgres()
                  ,user="stockmarketreader"
                  ,password="read123"
                  ,host="localhost"
                  ,port=5432
                  ,dbname="stockmarket"
```

```
)
#step 2:grab customer calendar, eod prices and indices and union eod
sources together-----------------------------------------------------

qry<-'SELECT * FROM custom_calendar ORDER by date'
ccal<-dbGetQuery(conn,qry)
#eod prices and indices
qry1="SELECT symbol,date,adj_close FROM eod_indices WHERE date BETWEEN
'2015-12-31' AND '2021-03-26'"
qry2="SELECT ticker,date,adj_close FROM eod_quotes WHERE date BETWEEN
'2015-12-31' AND '2021-03-26'AND (ticker = 'SLF' or ticker = 'FBC' or
ticker = 'ZNGA' or ticker = 'WM' or ticker = 'MLM' or ticker = 'HD' or
ticker = 'CSOD' or ticker = 'IRM' or ticker = 'KMB' or ticker = 'PZN'
or ticker = 'NP' or ticker = 'IDCC' or ticker = 'FELE' or ticker =
'GGB' or ticker = 'FAST')"
eod<-dbGetQuery(conn,paste(qry1,'UNION',qry2))
dbDisconnect(conn)
rm(conn)

#checks
head(ccal)
tail(ccal)
nrow(ccal)

head(eod)
tail(eod)
nrow(eod)

#For monthly we add two more data items (for 2015-12-31 & 2021-03-26)
eod_row<-data.frame(symbol='SP500TR',date=as.Date('2015-12-
31'),adj_close=3821.60)
eod_row_2<-data.frame(symbol = 'SP500TR', date=as.Date('2021-03-26'),
adj_close=8240.38)
eod<-rbind(eod,eod_row)
eod<-rbind(eod,eod_row_2)
#check
tail(eod)

# step 3: use calendar-------------------------------------------------
tday<-ccal[which(ccal$trading==1),,drop=F] #selecting only trading
days
head(tday)
tail(tday)
nrow(tday)-1 #trading days between 2016 and 2020 (excludes 12/31/2015)

#step 4: check completeness-------------------------------------------
#Percentage of completeness
pct<-table(eod$symbol)/(nrow(tday)-1)
```

```
selected_symbols_monthly<-names(pct)[which(pct>=0.99)]
eod_complete<-eod[which(eod$symbol %in%
selected_symbols_monthly),,drop=F]

#check
head(eod_complete)
tail(eod_complete)
nrow(eod_complete)

#step 5: transform (pivot)----------------------------------------
require(reshape2)
eod_pvt<-dcast(eod_complete, date ~
symbol,value.var='adj_close',fun.aggregate = mean, fill=NULL)

#check
eod_pvt[1:10,1:16] #first 10 rows and all columns
ncol(eod_pvt) # column count
nrow(eod_pvt)
head(eod_pvt)
tail(eod_pvt)

#step 6: merge with calendar--------------------------------------
eod_pvt_complete<-
merge.data.frame(x=tday[,'date',drop=F],y=eod_pvt,by='date',all.x=T)

#check
eod_pvt_complete[1:10,1:17] #first 10 rows and all columns
ncol(eod_pvt_complete)
nrow(eod_pvt_complete)
head(eod_pvt_complete)
tail(eod_pvt_complete)

#use dates as row names and remove the date column
rownames(eod_pvt_complete)<-eod_pvt_complete$date
eod_pvt_complete$date<-NULL #remove the "date" column

#re-check
eod_pvt_complete[1:10,1:5] #first 10 rows and first 5 columns
ncol(eod_pvt_complete)
nrow(eod_pvt_complete)

#step 7: missing data imputation----------------------------------
require(zoo)
eod_pvt_complete<-
na.locf(eod_pvt_complete,na.rm=F,fromLast=F,maxgap=3)

#re-check
```

```
eod_pvt_complete[1:10,1:16] #first 10 rows and first 5 columns
ncol(eod_pvt_complete)
nrow(eod_pvt_complete)

#step 8: calculating returns----------------------------------------
require(PerformanceAnalytics)
eod_ret<-CalculateReturns(eod_pvt_complete)

#check
eod_ret[1:10,1:3] #first 10 rows and first 3 columns
ncol(eod_ret)
nrow(eod_ret)

#remove the first row
eod_ret<-tail(eod_ret,-1) #use tail with a negative value

#check eod_ret[1:10,1:3] #first 10 rows and first 3 columns
ncol(eod_ret)
nrow(eod_ret)

#step 9: check for extreme returns----------------------------------
colMax <- function(data) sapply(data, max, na.rm = TRUE)
#apply it
max_daily_ret<-colMax(eod_ret) max_daily_ret[1:10] #first 10 max
returns
selected_symbols_daily<-
names(max_daily_ret)[which(max_daily_ret<=1.00)]
length(selected_symbols_daily)

#subset eod_ret
eod_ret<-eod_ret[,which(colnames(eod_ret) %in%
selected_symbols_daily),drop=F]

#check
eod_ret[1:10,1:16] #first 10 rows and first 3 columns
ncol(eod_ret)
nrow(eod_ret)

#step 10: tabular return data analytics-----------------------------
Ra<-
as.xts(eod_ret[,c('CSOD','FAST','FBC','FELE','GGB','HD','IDCC','IRM','
KMB', 'MLM','NP','PZN','SLF','WM','ZNGA'),drop=F])
Rb<-as.xts(eod_ret[,'SP500TR',drop=F]) #benchmark

#checks
head(Ra)
head(Rb)
tail(Rb)
```

```
#Returns
table.AnnualizedReturns(cbind(Rb,Ra),scale=252)

#Accumulate Returns
acc_Ra<-Return.cumulative(Ra);acc_Ra
acc_Rb<-Return.cumulative(Rb);acc_Rb

#Cumulative returns chart
chart.CumReturns(Ra,legend.loc = 'topleft')
chart.CumReturns(Rb,legend.loc = 'topleft')

#step 11: MV portfolio optimization ---------------------------------
#withhold the last 58 trading days excludes 2021 dates we have
Ra_training<-head(Ra,-58)
Rb_training<-head(Rb,-58)

#check
tail(Ra_training)
head(Ra_training)

#use the last 253 trading days for testing
Ra_testing<-tail(Ra,58)
Rb_testing<-tail(Rb,58)

#check
head(Ra_testing)
tail(Ra_testing)
tail(Ra_testing)

#optimize the MV (Markowitz 1950s) portfolio weights based on training
table.AnnualizedReturns(Rb_training) #annual minimum acceptable return
mar<-mean(Rb_training) #daily minimum acceptable return

require(PortfolioAnalytics)
require(ROI)
require(ROI.plugin.quadprog)
pspec<-portfolio.spec(assets=colnames(Ra_training))
pspec<-add.objective(portfolio=pspec,type="risk",name='StdDev')
pspec<-add.constraint(portfolio=pspec,type="full_investment")
pspec<-add.constraint(portfolio=pspec,type="return",return_target=mar)




#optimize portfolio
```

```
opt_p<-
optimize.portfolio(R=Ra_training,portfolio=pspec,optimize_method =
'ROI')

#extract weights (negative weights means shorting)
opt_w<-opt_p$weights

#apply weights to test returns
Rp<-Rb_testing
Rp$ptf<-Ra_testing %*% opt_w

#check
head(Rp)
tail(Rp)
#Compare basic metrics
table.AnnualizedReturns(Rp)

#step 12: chart hypothetical portfolio returns ----------------------
chart.CumReturns(Rp,legend.loc = 'bottomright')
```