ISDS 570 Group Project
Mean-Variance (MV) Portfolio Optimization

Team 5

Aashi Jitendra Rathod

Anuhya Ramaraju

Shubham Mandhare

Vishnu Sai Tharun Kumar Uppugunduri

Professor:

Dr. Pawel Kalczynski

May 14, 2023

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The project focuses on analyzing the returns of 12 companies whose names begin with the last name initials of the team members. The analysis uses data from the year 2016 to 2020 and we forecast the prices for 2021-01-01 to 2021-03-26. The project follows the ETL (Extract, Transform, Load) process to integrate data.

The data extraction step involves gathering relevant stock data from reliable sources, ensuring the dataset's accuracy and reliability. The custom calendar is manually created, which includes holidays, weekends, and non-trading days, providing a structured framework for accurate performance analysis. The project conducts data analysis and calculates returns for the selected tickers and the S&P 500 index, providing insights into their performance and relationship. Cumulative returns charts are generated for visualization purposes, allowing for a comparison of individual tickers to the benchmark.

Additionally, the project optimizes the portfolio using the mean-variance portfolio optimization method also known as the MV (Markowitz 1950s) portfolio method. The optimization aims to minimize risk while achieving the minimum acceptable return. The performance of the optimized portfolio is evaluated and compared to the benchmark index. The optimized portfolio has an annualized return of 45.76% outperforming the benchmark index which has an annualized return of 29.87%. However, the optimized portfolio does exhibit a higher volatility (higher risk) as compared to the S&P 500 index.

The tickers used the portfolio are as follows:

Tickers
MSFT, MERC, MSN, RES, RNR.PE, ROG, RADA, RPAI, RVNC, UBA, USAT, UEIC.

Overall, the project provides valuable insights into the performance of the selected tickers and their relationship to the broader market, enabling better investment decision-making.

1. ETL PROCESS

ETL stands for Extract, Transform and Load, a process which is used for data integration to move data from multiple sources into the targeted system. It is an essential step in data warehousing, business intelligence and big data analytics.

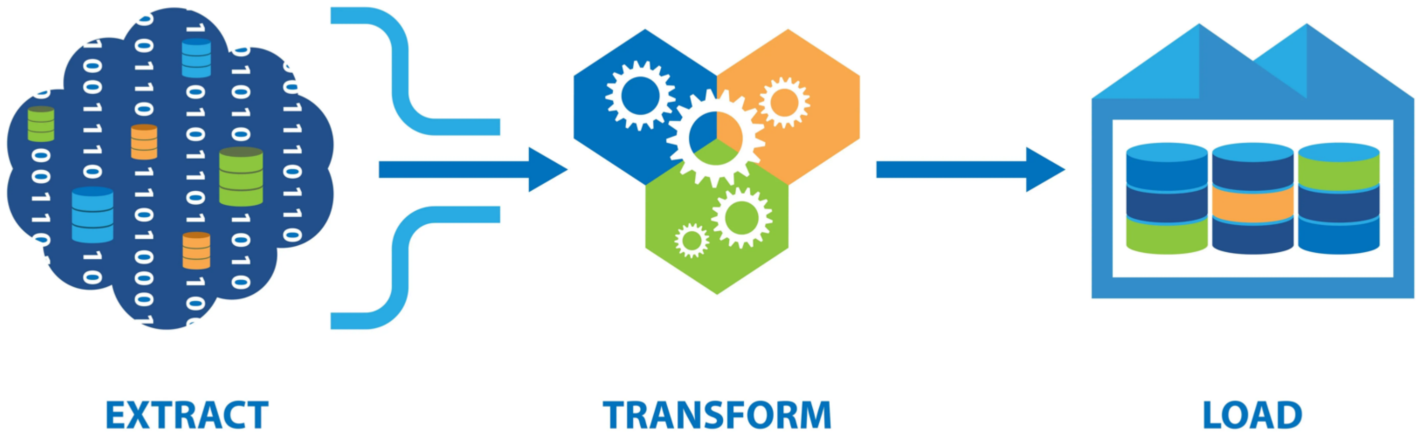

Figure 1: ETL Process

In the Extract step, relevant data is found and extracted from a variety of sources, including stock exchanges, and financial databases. To maintain consistency and compatibility, the data must be cleaned and standardized at the transform stage. This might include cleaning out duplicate records, fixing mistakes, and formatting data consistently. During the Load stage, the transformed data is loaded into the target system.

ETL enables analysts to see trends and patterns in stock market data that might not be obvious from just one source by combining data from various sources.

In conclusion, ETL is a procedure that makes it possible to take data from many sources, transform it into a standard format, and then load that format into a destination system for analysis.

For the stock market data, ETL plays a critical role in consolidating data from the sources and preparing it for analysis, ultimately enabling better investment decisions.
The goal of our project is to analyze the returns of 12 companies whose names begin with the initials of our team members' names. We will be examining data from the years 2016 to 2020 to determine these returns.

a) Extraction:

We extracted stock data from two reliable sources, Yahoo Finance, and Quandl, covering the period from January 1, 2016, to March 26, 2021. These sources provide comprehensive and accurate stock market information, enabling us to obtain a reliable dataset for our analysis.

Fetching data for SP500TR Index:
When evaluating investment performance in the American stock market, performance and portfolio analytics play a vital role. The SP500TR serves as a widely recognized benchmark for assessing investment performance in the American stock market. The index consists of 500 large-cap U.S. stocks spanning different sectors.

Custom Calendar:
To create our custom calendar, we manually list holidays and non-trading days (weekends and Presidents Day). We do this by using the function NETWORKDAYS.INTL. In addition to weekends, there are three market holidays that need to be considered: New Year's Day, which falls on January 1, 2021; Martin Luther King Jr. Day, observed on January 18, 2021; and Presidents Day, observed on February 15, 2021.

We generate a comprehensive custom calendar by organizing this information in a structured format, including the day, month, year, and trading/non-trading status. This calendar meticulously accommodates holidays, weekends, and other non-trading days, providing a reliable framework for our accurate performance analysis.

b) Transformation:

To calculate statistics such as the minimum, average, maximum, and volatility (standard deviation) of prices or index values for each month, we performed a join operation between the custom calendar table and the "v_eod_quotes_2016_2020" table. This allowed us to align the trading days and corresponding prices/index values for analysis. This analysis revealed that some tickers needed values, indicating incomplete data for certain trading days.

To identify tickers with complete trading day prices (99% or more), we calculated the percentage of complete trading day prices for each stock. This involved comparing the count of price items for each ticker against the total count of trading days within the specified date range. We filtered the results to include only tickers with a 99% or higher completeness percentage.

This analysis allowed us to identify the tickers with a high percentage of complete trading day prices, indicating more reliable and comprehensive data for further analysis.

To ensure the completeness and accuracy of our data, we implemented a process to exclude tickers with less than 99% complete trading day prices. By dividing the number of observations per ticker by the total number of observations and comparing it to 99%, we identified the tickers to be excluded. We created a table named "Exclusions_2016_2020" to store these excluded tickers for further analysis.

To improve the execution speed of our code, as it took a long time to retrieve the data, we implemented a materialized view. This view needed to be refreshed using the REFRESH function whenever there were changes in the underlying data.

Transformation in R:

The process starts by combining the end of day eod data for the S&P 500 index from the previous day with the eod data frame for trading days between 2016 to 2020. This is done by appending the S&P 500 data to the eod data frame using the values and rbind functions. A completeness score is then calculated for each stock ticker based on the number of days it was traded.

Only tickers with a completeness score greater than 99% are included in the new data set, eod_complete. The eod_complete data is then pivoted using the dcast function from the reshape 2 package, with the date as the rows, the symbol as the columns, and the adjusted closing price as the values. The calendar is then merged with the eod_complete data using the merge.data.frame function to ensure that no trading days were missed.

To create an extensible time series, the date column is removed from the data set and assigned as the row names using the rownames function. The extra date column is then removed by assigning null values. To avoid missing values, the na.locf function from the zoo package is used to carry forward the last observation for a maximum gap of 3.

c) Loading:

Subsequently, we pivoted the exported data in R for further analysis and visualization. We make a connection of Postgresql with R using Rpostgres and DBI package.

Calculating Returns:

Returns are then calculated using the Performance Analytics package and the Calculate Returns function. The first day's returns are calculated using the last trading date of the previous year. The extreme returns are checked using the colMax function to find the maximum returns for each column. Returns greater than 100% are removed from the data set using a subset function.

Our group performed tabular return data analytics using the S&P 500 index and 12 tickers that we selected based on the last names of the group members. The tickers we chose were "MSFT", "MERC", "MSN", "RES", "RNR.PE", "ROG", "RADA", "RPAI", "RVNC", "UBA", "USAT", and "UEIC".

We used the set.seed() function to reproduce the same results.

To convert the data into an extensible time series, we used the xts package in R. We assigned the returns of the 12 chosen tickers to Ra and the return of the S&P 500 index to Rb. To calculate the annualized returns, we used the table.AnnualizedReturns function. To calculate the cumulative returns, we used the Return.Cumulative function.

We also generated cumulative returns charts for both the tickers and the S&P 500 index using the chart. CumReturns function. These charts allow us to visualize the performance of each ticker compared to the benchmark over time.

Overall, our analysis provides valuable insights into the performance of the 12 tickers we selected and their relationship to the broader market represented by the S&P 500 index.

2. OPTIMIZATION OF PORTFOLIO

The portfolio is optimized using the traditional mean-variance portfolio optimization method. The optimization is performed using the training data, which includes all data up to the last 58 trading days. The last 58 trading days are reserved for testing the optimized portfolio.

First, the training data is extracted by using the head() function to include all data except for the last 58 trading days which contains the trading days from January 1, 2021 - March 26, 2021. Similarly, the testing data is extracted using the tail() function to include only the last 58 trading days.

Next, the minimum acceptable return is determined by calculating the mean return of the benchmark (SP500TR) for the training data. This minimum return value is assigned to the variable mar.

The PortfolioAnalytics package is used to optimize the portfolio using the mean-variance optimization method. To do this, a portfolio specification is first created using the portfolio.spec() function and providing the names of the assets (tickers).

Next, objectives and constraints are added to the portfolio specification. The objective is to minimize risk, which is done by setting the optimize_method argument to "ROI" and using the add.objective() function to specify the risk measure to be used. The constraints are added using the add.constraint() function. The first constraint added is to ensure full investment, which is done by setting the type of constraint to "full_investment". The second constraint added is to ensure that the portfolio achieves the minimum acceptable return mar.

Finally, the optimization is performed using the optimize.portfolio() function, which takes the training data, portfolio specification, and optimization method as inputs. The result of the optimization is a vector of weights for each asset in the portfolio, which is stored in the opt_w data frame. The round () function is used to display the weights with four digits of precision, and the sum() function is used to ensure that the weights sum to 1, indicating that the entire portfolio is fully invested.

Overall, this optimization process aims to find the combination of asset weights that minimizes portfolio risk while achieving the minimum acceptable return, as determined by the mean return of the benchmark for the training data.

3. PERFORMANCE

a) Cumulative return chart for individual stocks (2016-2020)

We have selected following tickers "MSFT", "MERC", "MSN", "RES", "RNR.PE", "ROG", "RADA", "RPAI", "RVNC", "UBA", "USAT","UEIC"

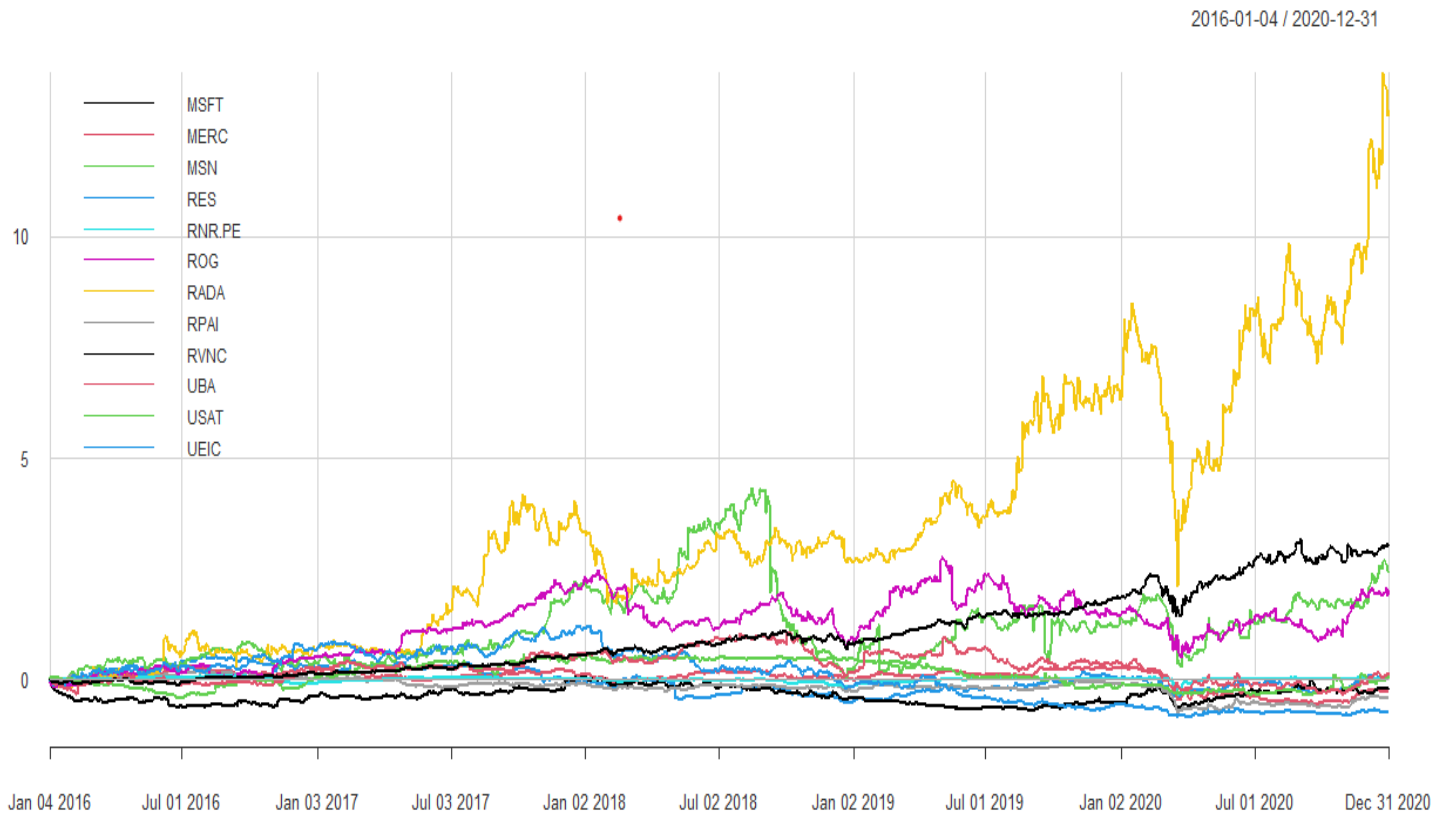

Figure 2: Cumulative Returns Chart

The above graph describes a cumulative returns chart that presents the performance of a set of selected ticker symbols from January 01, 2016, to December 31, 2020.The snippet highlights that the ticker symbol 'RADA' experienced a consistent upward trend in its cumulative returns by the end of December 31, 2020. This indicates that the value of the investment represented by the 'RADA' ticker steadily increased during the mentioned time frame.

Furthermore, the statement observes that almost all the ticker symbols experienced a drop around March-April 2020 due to the COVID-19 pandemic. This observation is in line with the general trend in the financial markets during that time, where the global pandemic caused a significant decline in the stock market as investors became increasingly risk-averse and companies experienced economic disruptions.

b) Weights of optimized portfolio

MSFT	**MERC**	**MSN**	**RES**
0.2698	0.0542	0.1021	-0.0084
RNR.PE	**ROG**	**RADA**	**RPAI**
0.5599	0.0170	0.0272	-0.0063
RVNC	**UBA**	**USAT**	**UEIC**
-0.0023	-0.0758	0.0215	0.0411

Table 1. Optimized weights and sum

The table 1 presents the weights assigned to various tickers in an optimized portfolio and the total sum of these weights. These weights indicate the proportion of the portfolio's total value allocated to each ticker. It is observed that specific tickers such as 'RES', 'RPAI', 'RVNC', and 'UBA' have a negative impact on the portfolio. This suggests that these tickers have had a detrimental effect on the portfolio's overall performance during the optimization process.

On the other hand, tickers like 'RNR.PE,' 'MSFT,' and 'MSN' hold significant investment percentages within the portfolio. This suggests that the optimization process aims to allocate a significant portion of the portfolio's value to these tickers. Therefore, it can be inferred that the optimization process seeks to maximize the overall returns while minimizing the risk associated with the negative-performing tickers.

Sum(weights)
1

Table 2. Sum of Weights

Furthermore, the table 2 specifies that the sum of all the weights equals 1, which is a fundamental requirement for a portfolio. This indicates that all the investment weights of the portfolio have been appropriately assigned to achieve the desired level of diversification and risk management.

c) Cumulative return chart of portfolio and SP500TR index (2021)

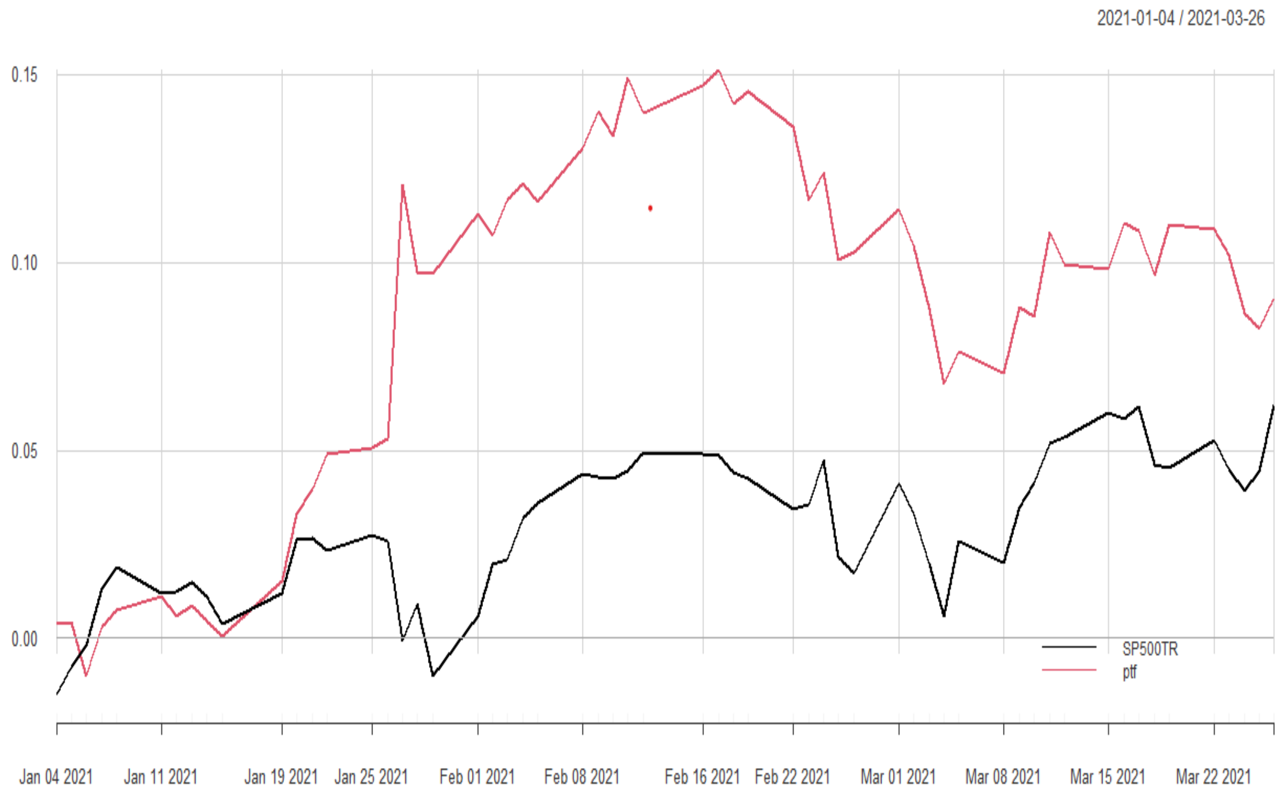

Figure 3: Portfolio vs SP500TR

The graph displays the portfolio performance and the SP500TR index over several weeks. In the initial three weeks of January, the portfolio closely followed the index, indicating a correlation between the portfolio's returns and the broader market. However, in the final week of January, the portfolio experienced significant growth while the index reached an all-time low, demonstrating the portfolio's outperformance. February showed steady growth for both the portfolio and the index, reflecting a favorable market environment. March started with a decline for both, but they began recovering in the second week, indicating market volatility.

Overall, the portfolio exhibited alignment with the market in January, exceptional growth in the last week, steady growth in February, and some volatility in March.

d) Annualized returns for portfolio and SP500TR index (2021)

As can be seen in the Table 4, our optimized portfolio has a better return. However, it carries higher risk too as the sharpe ratio is higher for the portfolio.

	SP500TR	Portfolio
Annualized Return	0.2987	0.4576
Annualized Standard Deviation	0.1623	0.2067
Annualized Sharp	1.8399	2.213

Table 3: Table of Annualized Return

e) Performance Comparison: Optimized Portfolio vs Index

Our optimized portfolio had a higher annualized return of 45.76% compared to the SP500TR index's return of 29.87%, resulting in a percentage increase of approximately 53.19%. The portfolio also had a slightly higher standard deviation of 0.2067 than the index's 0.1623, indicating a higher risk. Furthermore, the portfolio has a higher Sharpe ratio of 2.2134 compared to the index's 1.8399, suggesting better risk-adjusted performance. Essentially, we assumed an additional risk of 0.3735 to earn a 15.89% higher return. Furthermore, when analyzing the cumulative return chart, we observed that our portfolio initially tracked the index but continued to grow while the index declined. Particularly, in the final week of January, our portfolio experienced significant growth while the index reached an all-time low, demonstrating its outperformance. Although both the portfolio and index experienced a decline in March, they started recovering in the second week. In conclusion, our optimized portfolio, despite being more volatile, delivered superior returns compared to the benchmark, showcasing its effectiveness in maximizing returns.

4. APPENDIX

SQL Code

Create Custom Calendar table and import the values

```
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;

-- Let's add some columns to be used later: 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;


-- 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;
-- 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;
-- We could use 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;
```

R Code

```
# Connect to PostgreSQL ----------------------------------------
----------

require(RPostgres) # did you install this package?
require(DBI)
conn <- dbConnect(RPostgres::Postgres()
                 ,user="stockmarketreader"
                 ,password="read123"
                 ,host="localhost"
                 ,port=5432
                 ,dbname="stockmarket"
)

#custom calendar
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'"
eod<-dbGetQuery(conn,paste(qry1,'UNION',qry2))
dbDisconnect(conn)
rm(conn)

#Explore
head(ccal)
tail(ccal)
nrow(ccal)

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

head(eod[which(eod$symbol=='SP500TR'),])

#For monthly we may need one more data item (for 2015-12-31)
#We can add it to the database (INSERT INTO) - but to practice:
eod_row<-data.frame(symbol='SP500TR',date=as.Date('2015-12-
31'),adj_close=3821.60) #3821.60
eod<-rbind(eod,eod_row)
tail(eod)

# Use Calendar --------------------------------------------------
------

tdays<-ccal[which(ccal$trading==1),,drop=F]
head(tdays)
nrow(tdays)-1 #trading days between 2016 and 2020

# Completeness --------------------------------------------------
--------
# Percentage of completeness
pct<-table(eod$symbol)/(nrow(tdays)-1)
selected_symbols_daily<-names(pct)[which(pct>=0.99)]
eod_complete<-eod[which(eod$symbol                          %in%
selected_symbols_daily),,drop=F]

#check
head(eod_complete)
```

```
tail(eod_complete)
nrow(eod_complete)

# Transform (Pivot) ----------------------------------------------------
----------

require(reshape2)
eod_pvt<-dcast(eod_complete,                date                 ~
symbol,value.var='adj_close',fun.aggregate = mean, fill=NULL)
#check
eod_pvt[1:10,1:5] #first 10 rows and first 5 columns
ncol(eod_pvt) # column count
nrow(eod_pvt)

# Merge with Calendar --------------------------------------------------
----------
eod_pvt_complete<-
merge.data.frame(x=tdays[,'date',drop=F],y=eod_pvt,by='date',all
.x=T)

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

# Missing Data Imputation ----------------------------------------------
--------------
# We can replace a few missing (NA or NaN) data items with previous
data
require(zoo)
eod_pvt_complete<-
na.locf(eod_pvt_complete,na.rm=F,fromLast=F,maxgap=3)
```

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

# 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)

# Check for extreme returns -------------------------------------
------
# There is colSums, colMeans but no colMax so we need to create it
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
# And proceed just like we did with percentage (completeness)
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[1:10,1:3] #first 10 rows and first 3 columns
ncol(eod_ret)
nrow(eod_ret)
```

```
# subset eom_ret data to exclude the unwanted rows
eod_ret = eod_ret[1:1317,1:4215]


# Tabular Return Data Analytics -----------------------------------
----------

# We will select 'SP500TR' and 12 TICKERS (3 for each student)
set.seed(1010)  #  seed  can  be  any  number,  it  will  ensure
repeatability
#random12 <- sample(colnames(eod_ret),12) #
random12  <-  c("MSFT",  "MERC",  "MSN",  "RES",  "RNR.PE",  "ROG",
"RADA", "RPAI", "RVNC", "UBA", "USAT","UEIC")
# We need to convert data frames to xts (extensible time series)
Ra<-as.xts(eod_ret[,random12,drop=F])
Rb<-as.xts(eod_ret[,'SP500TR',drop=F]) #benchmark

head(Ra)
head(Rb)

# And now we can use the analytical package...

# Stats
table.Stats(Ra)

# Distributions
table.Distributions(Ra)

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

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

# Capital Assets Pricing Model
table.CAPM(Ra,Rb)

# Graphical Return Data Analytics ---------------------------------
----------
```

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

#Box plots
chart.Boxplot(cbind(Rb,Ra))

chart.Drawdown(Ra,legend.loc = 'bottomleft')

# MV Portfolio Optimization -------------------------------------
----------

# withhold the last 58 trading days (the days that we have to
predict)
Ra_training<-head(Ra,-58)
Rb_training<-head(Rb,-58)

head(Ra_training)
tail(Ra_training)

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

tail(Ra_testing)
head(Ra_testing)

#optimize the MV (Markowitz 1950s) portfolio weights based on
training
table.AnnualizedReturns(Rb_training)
mar<-mean(Rb_training) #we need 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

round(opt_w,4)
sum(opt_w)

#apply weights to test returns
Rp<-Rb_testing # easier to apply the existing structure
#define new column that is the dot product of the two vectors
Rp$ptf<-Ra_testing %*% opt_w

#check
head(Rp)
tail(Rp)

#Compare basic metrics
table.AnnualizedReturns(Rp)

# Chart Hypothetical Portfolio Returns --------------------------
----------

chart.CumReturns(Rp,legend.loc          =          'bottomright')
#Return.cumulative(Rp)

# End of Stock Market Case Study Project
```