ISDS 570 Group Project (MV Portfolio Optimization)

Group Members: Christina Lozano, Gregory Gonzalez, Harout Krboyan, Timothy Lue

Instructor: Dr. Pawel J. Kalczynski

Course: ISDS 570-59 Business Data Transformation

Date: June 30, 2022

Introduction

Our team will be optimizing a mean-variance portfolio using five years of daily data ranging from 2016 to 2020 for the purpose of testing the portfolio for the first 3 months of 2021. For our optimization, we used the minimum acceptable return of the SP500TR index for our five years of daily data. We begin our presentation by diving into a more detailed account of our data as well as the ETL process.

Data & ETL

Data that is used within this project includes the csv files containing the company list and the eod csv file. After creating the company list table and importing the files, the eod_quotes table is created and imported with the eod csv file. Next, we create a view for eod_quotes called v_eod_quotes_2016_2021 setting the date range from 2016-01-01 to 2021-03-26. Then we create a table for SP500TR called eod_indices, which contains imported data that is scraped from Yahoo https://finance.yahoo.com/quote/%5ESP500TR/history?p=^SP500TR, with the same date ranges of 2016-01-01 to 2021-03-26. From there, another view is created for eod_indices with the same date range naming the view as v_eod_indices_2016_2021. A table for custom_calendar is created with the same date range while marking the trading days with "1" and non-trading days (weekends & holidays) with "0." To incorporate the end of the month (eom) and previous trading day (prev_trading_day), the table is updated with the following columns and queries are executed to obtain the necessary data. Furthermore, an exclusions_2016_2021 table is created which shows tickers that are less than 99% complete. Once that is created a new view is created, v_eod_2016_2021, which joins the v_eod_indices_2016_2021 and v_eod_quotes_2016_2021 table with exclusions. To simplify the process, a materialized view is created with the same query and is refreshed with the data called mv_eod_2016_2021. The same process for materialized view is then applied for returns by joining the custom calendar and mv_eod_2016_2021 previous trading day to mv_eod_2016_2021 (eod) creating the view mv_ret_2016_2021 with refreshing the data. Then we insert the returns that are higher than 100% into mv_ret_2016_2021 and refresh the materialized view with the new updates to the table. Lastly, we create a daily prices table when joining mv_eod_2016_2021 to custom_calendar called export_daily_prices_2016_2021, will be exported into a csv file and imported in RStudio for further analysis.

The ETL process begins with importing the daily_prices_2016_2021 csv file to RStudio creating a path to the file. The file should be displayed in RStudio, and our database is connected

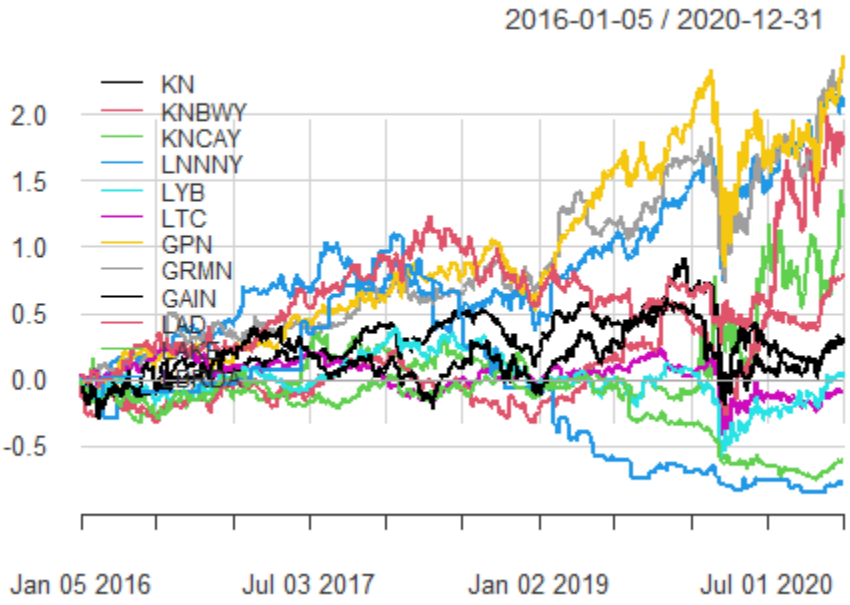
by creation of a user in PostgreSQL. This user allows us to access all the data within our database. We are then able to develop queries that can extract the data that we need to perform our analysis by extracting all from the custom_calendar and joining the eod_indices with eod_quotes from 2016-01-01 to 2021-03-26. Once we have all the data, we disconnect the database and can perform our analysis now that everything is in RStudio. The joined query produces the eod table, which is then used to get the eod_complete which encompasses the percentage of completeness. The eod_complete table is then pivoted and merged to the calendar that we have created. In addition, we remove the date column heading and require the package "zoo" to cleanse our table of missing data. Our new table is now called eod_pvt_complete. With this table created, we are now able to optimize our portfolio and calculate our cumulative and annual return for chosen stock ticker symbols and SP500TR.

Analysis

Cumulative Return for 12 Chosen Stock Tickers (2016-2020)

Chart 1 below depicts the cumulative returns for 2016-2020 for the tickers our team selected. From this chart, we can see that our team has selected a distribution of tickers that ends with most tickers above 0 cumulative returns at the end of 2020. It is interesting to note that our distribution of tickers has a high of nearly 2 cumulative returns with a low of under –0.5 cumulative returns in the year 2020. This means that the general trend of the cumulative returns for our team selected tickers is positive sloping with a positive ROI.

Chart 1



Weights of Optimized Portfolio

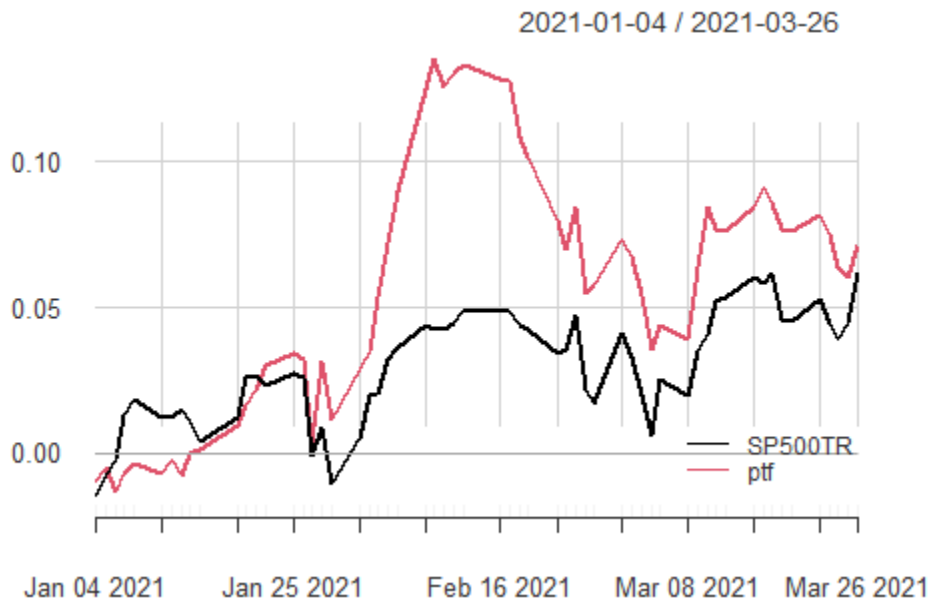
Table 1 below depicts the weights of our optimized portfolio to four digits of precision with the sum of the weights. From the table, we can see that the tickers like KNBWY and LBRDA have the highest weights and LYB has the lowest weight. The higher the weight, the larger the amount the ticker has in the distribution. Conversely, the lower the weight, the lower the amount the ticker has in the distribution. This means that KNBWY and LBRDA have large distributions, whereas LYB has small/negative distributions. The sum of the weights is 1 which means that our weights are properly normalized.

Table 1	Weights
GAIN	0.1523
GPN	0.0097
GRMN	0.1728
KN	0.0117
KNBWY	0.2409
KNCAY	0.0962
LAD	0.0141
LAKE	0.1223
LBRDA	0.1882
LNNNY	0.0571
LTC	0.0237
LYB	-0.0890
SUM	1

Cumulative Return Chart for Optimized Portfolio and SP500TR (2021)

Chart 2 below depicts the cumulative returns for our portfolio and SP500TR index for 2021 ending on March 26th. From this chart we can observe that our portfolio and the SP500TR index start at around the same number of cumulative returns, while our portfolio overtakes the SP500TR index near the end of January. At its peak, our portfolio had an advantage of around 0.05 cumulative returns in the month of February. Even after February, our portfolio is still consistently above the SP500TR index. Thus, we can determine that our portfolio appears to have a higher cumulative return on average compared to the SP500TR index.

Chart 2



Annualized Returns for Optimized Portfolio and SP500TR

Table 2 below contains the annualized returns for our portfolio and SP500TR index for 2021 ending on March 26th. From this table we observe that the annualized return for our portfolio is approximately 0.3479 with a standard deviation of 0.1983. The annualized return for the SP500TR index is approximately 0.2987 with a standard deviation of 0.1623. This means that our portfolio has a higher annualized return with a larger variance in returns, while the SP500TR index has a lower annualized return with a smaller variance in returns.

Table 2	SP500TR	ptf
Annualized Return	0.2987	0.3479
Annualized Std Dev	0.1623	0.1983
Annualized Sharpe (Rf=0%)	1.8399	1.7547

Conclusion

From the analysis above, we can determine that the performance of our optimized portfolio is superior to the performance of the SP500TR index. This conclusion is reached through comparing the annualized returns for our portfolio and the SP500TR index alongside the charts generated from the cumulative returns. From Table 2, we can determine that our portfolio has a higher annualized return with a larger variance in returns than the SP500TR index. By

looking at Chart 2, we can observe that our portfolio appears to have a higher cumulative return during the three-month period of 2021 compared to the SP500TR index. Through the direct numerical and visual comparisons of our analysis, we can speculate that exploring custom selection of stocks may be more profitable compared to investing in the totality of the SP500TR index.

Appendix

PostgreSQL

```
--ISDS 570 Group Project
--Create eod_quotes; import eod.csv
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;


--Create eod view 2016-2021
CREATE OR REPLACE VIEW public.v_eod_quotes_2016_2021 AS
SELECT eod_quotes.ticker,
    eod_quotes.date,
    eod_quotes.adj_close
   FROM eod_quotes
  WHERE eod_quotes.date >= '2016-01-01'::date AND eod_quotes.date <= '2021-03-26'::date;

ALTER TABLE public.v_eod_quotes_2016_2021
    OWNER TO postgres;
```

```
--Create eod_indices table; import data from 2016-01-01 to 2021-03-26
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;


--Create eod_indices view 2016-2021
CREATE OR REPLACE VIEW public.v_eod_indices_2016_2021 AS
SELECT eod_indices.symbol,
    eod_indices.date,
    eod_indices.adj_close
   FROM eod_indices
   WHERE eod_indices.date >= '2016-01-01'::date AND eod_indices.date <= '2021-03-
26'::date;


ALTER TABLE public.v_eod_indices_2016_2021
    OWNER TO postgres;

--Create custom calendar; import custom calendar with 2016-2021 data
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;

--Alter table with eom and prev trading day
ALTER TABLE public.custom_calendar
    ADD COLUMN eom smallint;

ALTER TABLE public.custom_calendar
    ADD COLUMN prev_trading_day date;

--Update prev trading day
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;

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

--Update eom
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;



-- Store the excluded tickers (less than 99% complete in a table)
SELECT ticker, 'More than 1% missing' as reason
INTO exclusions_2016_2021
FROM v_eod_quotes_2016_2021
```

```
GROUP BY ticker
HAVING count(*)::real/(SELECT COUNT(*) FROM custom_calendar WHERE trading=1
AND date BETWEEN '2016-01-01' AND '2021-03-26')::real<0.99;

ALTER TABLE public.exclusions_2016_2021
    ADD CONSTRAINT exclusions_2016_2021_pkey PRIMARY KEY (ticker);

--Create view
CREATE OR REPLACE VIEW public.v_eod_2016_2021 AS
SELECT v_eod_indices_2016_2021.symbol,
    v_eod_indices_2016_2021.date,
    v_eod_indices_2016_2021.adj_close
   FROM v_eod_indices_2016_2021
  WHERE NOT (v_eod_indices_2016_2021.symbol::text IN ( SELECT DISTINCT
exclusions_2016_2021.ticker
          FROM exclusions_2016_2021))
UNION
SELECT v_eod_quotes_2016_2021.ticker AS symbol,
    v_eod_quotes_2016_2021.date,
    v_eod_quotes_2016_2021.adj_close
   FROM v_eod_quotes_2016_2021
  WHERE NOT (v_eod_quotes_2016_2021.ticker::text IN ( SELECT DISTINCT
exclusions_2016_2021.ticker
          FROM exclusions_2016_2021));

ALTER TABLE public.v_eod_2016_2021
    OWNER TO postgres;

--Create eod materialized view
CREATE MATERIALIZED VIEW public.mv_eod_2016_2021
TABLESPACE pg_default
AS
SELECT v_eod_indices_2016_2021.symbol,
    v_eod_indices_2016_2021.date,
    v_eod_indices_2016_2021.adj_close
   FROM v_eod_indices_2016_2021
  WHERE NOT (v_eod_indices_2016_2021.symbol::text IN ( SELECT DISTINCT
exclusions_2016_2021.ticker
          FROM exclusions_2016_2021))
UNION
SELECT v_eod_quotes_2016_2021.ticker AS symbol,
    v_eod_quotes_2016_2021.date,
    v_eod_quotes_2016_2021.adj_close
```

```
   FROM v_eod_quotes_2016_2021
  WHERE NOT (v_eod_quotes_2016_2021.ticker::text IN ( SELECT DISTINCT
exclusions_2016_2021.ticker
         FROM exclusions_2016_2021))
WITH NO DATA;

ALTER TABLE public.mv_eod_2016_2021
    OWNER TO postgres;


--Refresh with data
REFRESH MATERIALIZED VIEW mv_eod_2016_2021 WITH DATA;

--Create ret materialized view
CREATE MATERIALIZED VIEW public.mv_ret_2016_2021
TABLESPACE pg_default
AS
SELECT eod.symbol,
    eod.date,
    eod.adj_close / prev_eod.adj_close - 1.0::double precision AS ret
   FROM mv_eod_2016_2021 eod
     JOIN custom_calendar cc ON eod.date = cc.date
     JOIN mv_eod_2016_2021 prev_eod ON prev_eod.symbol::text = eod.symbol::text AND
prev_eod.date = cc.prev_trading_day
WITH NO DATA;

ALTER TABLE public.mv_ret_2016_2021
    OWNER TO postgres;

--Refresh with data
REFRESH MATERIALIZED VIEW mv_ret_2016_2021 WITH DATA;

--Insert into exclusions
INSERT INTO exclusions_2016_2021
SELECT DISTINCT symbol, 'Return higher than 100%' as reason FROM mv_ret_2016_2021
WHERE ret>1.0;

--Refresh mv eod and mv ret
REFRESH MATERIALIZED VIEW mv_eod_2016_2021 WITH DATA;
REFRESH MATERIALIZED VIEW mv_ret_2016_2021 WITH DATA;

--Export Daily Prices 2016-2021
SELECT PR.*
INTO export_daily_prices_2016_2021
```

```
FROM custom_calendar CC LEFT JOIN mv_eod_2016_2021 PR ON CC.date=PR.date
WHERE CC.trading=1;


CREATE USER groupprojectreader 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 groupprojectreader;

-- Grant read rights (for future tables and views)
ALTER DEFAULT PRIVILEGES IN SCHEMA public
   GRANT SELECT ON TABLES TO groupprojectreader;
```

RStudio

```
#ISDS 570 Group Project
rm(list=ls(all=T)) # this just removes everything from memory
# Load CSV Files ---------------------------------------------------------

# Load daily prices from CSV - no parameters needed
dp<-read.csv('C:/Users/ggonz/Desktop/ISDS 570/ISDS 570 Group
Project/daily_prices_2016_2021.csv') # no arguments

#Explore
head(dp) #first few rows
tail(dp) #last few rows

dp<-head(dp,-1)



rm(dp) # remove from memory



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

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

#custom calendar
qry<-"SELECT * FROM custom_calendar WHERE date BETWEEN '2016-01-01' AND
'2021-03-26'"
ccal<-dbGetQuery(conn,qry)
#eod prices and indices
qry1="SELECT symbol,date,adj_close FROM eod_indices WHERE date BETWEEN '2016-
01-01' AND '2021-03-26'"
qry2="SELECT ticker,date,adj_close FROM eod_quotes WHERE date BETWEEN '2016-01-
01' 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$symbol=='SP500TR'),])
tail(eod[which(eod$symbol=='SP500TR'),])



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

tdays<-ccal[which(ccal$trading==1),,drop=F]
tail(tdays)
nrow(tdays)-1 #trading days between 2015 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 ----------------------------------------------
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)

# YOUR TURN: calculate eom_ret (monthly returns)

# 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)



# Export data from R to CSV -----------------------------------------
write.csv(eod_ret,'C:/Temp/eod_ret.csv')



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

# We will select 'SP500TR' and 12 RANDOM TICKERS

random12 <-
c('KN','KNBWY','KNCAY','LNNNY','LYB','LTC','GPN','GRMN','GAIN','LAD','LAKE','LBRDA')
```

```
random12

# 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)
tail(Ra)
head(Rb)
tail(Rb)



# Stats
table.Stats(Ra)

# Distributions
table.Distributions(Ra)

# Returns
table.AnnualizedReturns(cbind(Rb,Ra),scale=252) # note for monthly use scale=12

# 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(head(Ra,-58),legend.loc = 'topleft') #1

chart.CumReturns(cbind(tail(Rb,58),tail(Ra,58)),legend.loc = 'topleft') #3
#Box plots
chart.Boxplot(cbind(Rb,Ra))

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



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

# withhold the last 253 trading days
Ra_training<-head(Ra,-58)
```

```
tail(Ra_training)
Rb_training<-head(Rb,-58)
tail(Rb_training)
# use the last 253 trading days for testing
Ra_testing<-tail(Ra,58)
head(Ra_testing)
Rb_testing<-tail(Rb,58)
tail(Rb_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) # make sure to install it
require(ROI.plugin.quadprog)  # make sure to install it
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')
opt_p
#extract weights (negative weights means shorting)
opt_w<-opt_p$weights
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')
```