

































































































































































# Built using Python 3.7.4 # Load libraries

import pandas as pd

import pandas\_datareader.data as web import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns plt.style.use('ggplot') sns.set()

# SKIP IF ALREADY HAVE DATA

# Load data

start\_date = '1970-01-01'

end\_date = '2019-12-31'

symbols = ["WILL5000INDFC", "BAMLCC0A0CMTRIV", "GOLDPMGBD228NLBM", "CSUSHPINSA", "DGS5"]

sym\_names = ["stock", "bond", "gold", "realt", 'rfr'] filename = 'data\_port\_const.pkl'

try:

df = pd.read\_pickle(filename) print('Data loaded')

except FileNotFoundError: print("File not found") print("Loading data", 30\*"-")

data = web.DataReader(symbols, 'fred', start\_date, end\_date) data.columns = sym\_names

data\_mon = data.resample('M').last()

df = data\_mon.pct\_change()['1987':'2019'] dat = data\_mon.pct\_change()['1971':'2019']

## Simulation function class Port\_sim:

def calc\_sim(df, sims, cols): wts = np.zeros((sims, cols))

for i in range(sims):

a = np.random.uniform(0,1,cols) b = a/np.sum(a)

wts[i,] = b

mean\_ret = df.mean() port\_cov = df.cov()

port = np.zeros((sims, 2)) for i in range(sims):

port[i,0] = np.sum(wts[i,]\*mean\_ret)

port[i,1] = np.sqrt(np.dot(np.dot(wts[i,].T,port\_cov), wts[i,]))

sharpe = port[:,0]/port[:,1]\*np.sqrt(12) best\_port = port[np.where(sharpe == max(sharpe))] max\_sharpe = max(sharpe)

return port, wts, best\_port, sharpe, max\_sharpe

def calc\_sim\_lv(df, sims, cols):

wts = np.zeros(((cols-1)\*sims, cols)) count=0

for i in range(1,cols): for j in range(sims):

a = np.random.uniform(0,1,(cols-i+1)) b = a/np.sum(a)

c = np.random.choice(np.concatenate((b, np.zeros(i))),cols, replace=False) wts[count,] = c

count+=1 mean\_ret = df.mean()

port\_cov = df.cov()

port = np.zeros(((cols-1)\*sims, 2)) for i in range(sims):

port[i,0] = np.sum(wts[i,]\*mean\_ret)

port[i,1] = np.sqrt(np.dot(np.dot(wts[i,].T,port\_cov), wts[i,]))

sharpe = port[:,0]/port[:,1]\*np.sqrt(12) best\_port = port[np.where(sharpe == max(sharpe))] max\_sharpe = max(sharpe)

return port, wts, best\_port, sharpe, max\_sharpe

def graph\_sim(port, sharpe): plt.figure(figsize=(14,6))

plt.scatter(port[:,1]\*np.sqrt(12)\*100, port[:,0]\*1200, marker='.', c=sharpe, cmap='Blues') plt.colorbar(label='Sharpe ratio', orientation = 'vertical', shrink = 0.25) plt.title('Simulated portfolios', fontsize=20)

plt.xlabel('Risk (%)')

plt.ylabel('Return (%)') plt.show()

# Calculate returns and risk for longer period hist\_mu = dat['1971':'1991'].mean(axis=0) hist\_sigma = dat['1971':'1991'].std(axis=0)

# Run simulation based on historical figures np.random.seed(123)

sim1 = []

for i in range(1000): #np.random.normal(mu, sigma, obs)

a = np.random.normal(hist\_mu[0], hist\_sigma[0], 60) + np.random.normal(0, hist\_sigma[0], 60) b = np.random.normal(hist\_mu[1], hist\_sigma[1], 60) + np.random.normal(0, hist\_sigma[1], 60) c = np.random.normal(hist\_mu[2], hist\_sigma[2], 60) + np.random.normal(0, hist\_sigma[2], 60) d = np.random.normal(hist\_mu[3], hist\_sigma[3], 60) + np.random.normal(0, hist\_sigma[3], 60)

df1 = pd.DataFrame(np.array([a, b, c, d]).T) cov\_df1 = df1.cov()

sim1.append([df1, cov\_df1])

# create graph objects np.random.seed(123)

samp = np.random.randint(1, 1000, 4) graphs1 = []

for i in range(4):

port, \_, \_, sharpe, \_ = Port\_sim.calc\_sim(sim1[samp[i]][0], 1000, 4) graf = [port,sharpe]

graphs1.append(graf)

# Graph sample portfolios

fig, axes = plt.subplots(2, 2, figsize=(12,6)) for i, ax in enumerate(fig.axes):

ax.scatter(graphs1[i][0][:,1]\*np.sqrt(12)\*100, graphs1[i][0][:,0]\*1200, marker='.', c=graphs1[i][1], plt.show()

# create graph objects np.random.seed(123) graphs2 = []

for i in range(4):

port, \_, \_, sharpe, \_ = Port\_sim.calc\_sim\_lv(sim1[samp[i]][0], 1000, 4) graf = [port,sharpe]

graphs2.append(graf)

# Graph sample portfolios

fig, axes = plt.subplots(2, 2, figsize=(12,6))

for i, ax in enumerate(fig.axes):

ax.scatter(graphs2[i][0][:,1]\*np.sqrt(12)\*100, graphs2[i][0][:,0]\*1200, marker='.', c=graphs2[i][1], plt.show()

# Calculate probability of hitting risk-return constraints based on sample portfolos probs = []

for i in range(8): if i <= 3:

out = round(np.mean((graphs1[i][0][:,0] >= 0.07/12) & (graphs1[i][0][:,1] <= 0.1/np.sqrt(12))),2 probs.append(out)

else:

out = round(np.mean((graphs2[i-4][0][:,0] >= 0.07/12) & (graphs2[i-4][0][:,1] <= 0.1/np.sqrt(12) probs.append(out)

print(probs)

# Simulate portfolios from reteurn simulations def wt\_func(sims, cols):

wts = np.zeros(((cols-1)\*sims, cols)) count=0

for i in range(1,cols): for j in range(sims):

a = np.random.uniform(0,1,(cols-i+1)) b = a/np.sum(a)

c = np.random.choice(np.concatenate((b, np.zeros(i))),cols, replace=False) wts[count,] = c

count+=1 return wts

# Note this takes over 4min to run, substantially worse than the R version, which runs in under a minute

np.random.seed(123)

portfolios = np.zeros((1000, 3000, 2)) weights = np.zeros((1000,3000,4))

for i in range(1000):

wt\_mat = wt\_func(1000,4)

port\_ret = sim1[i][0].mean(axis=0) cov\_dat = sim1[i][0].cov()

returns = np.dot(wt\_mat, port\_ret)

risk = [np.sqrt(np.dot(np.dot(wt.T,cov\_dat), wt)) for wt in wt\_mat] portfolios[i][:,0] = returns

portfolios[i][:,1] = risk weights[i][:,:] = wt\_mat

port\_1m = portfolios.reshape((3000000,2)) wt\_1m = weights.reshape((3000000,4))

# Find probability of hitting risk-return constraints on simulated portfolios

port\_1m\_prob = round(np.mean((port\_1m[:][:,0] > 0.07/12) & (port\_1m[:][:,1] <= 0.1/np.sqrt(12))),2)\*100 print(f"The probability of meeting our portfolio constraints is:{port\_1m\_prob: 0.0f}%")

# Plot sample portfolios np.random.seed(123)

port\_samp = port\_1m[np.random.choice(1000000, 10000),:] sharpe = port\_samp[:,0]/port\_samp[:,1]

plt.figure(figsize=(14,6))

plt.scatter(port\_samp[:,1]\*np.sqrt(12)\*100, port\_samp[:,0]\*1200, marker='.', c=sharpe, cmap='Blues') plt.colorbar(label='Sharpe ratio', orientation = 'vertical', shrink = 0.25)

plt.title('Ten thousand samples from three million simulated portfolios', fontsize=20) plt.xlabel('Risk (%)')

plt.ylabel('Return (%)') plt.show()

# Graph histograms

fig, axes = plt.subplots(1,2, figsize = (12,6))

for idx,ax in enumerate(fig.axes): if idx == 1:

ax.hist(port\_1m[:][:,1], bins = 100) else:

ax.hist(port\_1m[:][:,0], bins = 100) plt.show()

## Create buckets for analysis and graphin

df\_port = pd.DataFrame(port\_1m, columns = ['returns', 'risk']) port\_bins = np.arange(-35,65,10)

df\_port['dig\_ret'] = pd.cut(df\_port['returns']\*1200, port\_bins)

xs = ["(-35, -25]", "(-25, -15]", "(-15, -5]","(-5, 5]", "(5, 15]", "(15, 25]", "(25, 35]", "(35, 45]",

ys = df\_port.groupby('dig\_ret').size().values/len(df\_port)\*100

# Graph buckets with frequency

fig,ax = plt.subplots(figsize = (12,6)) ax.bar(xs[2:7], ys[2:7])

ax.set(xlabel = "Return bucket (%)", ylabel = "Frequency (%)",

title = "Frequency of occurrrence for return bucket ") plt.show()

# Calculate frequency of occurence for mid range of returns

good\_range = np.sum(df\_port.groupby('dig\_ret').size()[4:6])/len(df\_port) good\_range

## Graph buckets with median return and risk

med\_ret = df\_port.groupby('dig\_ret').agg({'returns':'median'})\*1200

med\_risk = df\_port.groupby('dig\_ret').agg({'risk':'median'})\*np.sqrt(12)\*100

labs\_ret = np.round(med\_ret['returns'].to\_list()[2:7]) labs\_risk = np.round(med\_risk['risk'].to\_list()[2:7])

fig, ax = plt.subplots(figsize = (12,6)) ax.bar(xs[2:7], ys[2:7])

for i in range(len(xs[2:7])):

ax.annotate(str('Returns: ' + str(labs\_ret[i])), xy = (xs[2:7][i], ys[2:7][i]+2), xycoords = 'data')

ax.annotate(str('Risk: ' + str(labs\_risk[i])), xy = (xs[2:7][i], ys[2:7][i]+5), xycoords = 'data')

ax.set(xlabel = "Return bucket (%)", ylabel = "Frequency (%)",

title = "Frequency of occurrrence for return bucket ", ylim = (0,60))

plt.show()

# Find frequency of high return buckets

hi\_range = np.sum(df\_port.groupby('dig\_ret').size()[6:])/len(df\_port) hi\_range

## Identify weights for different buckets for graphing

wt\_1m = pd.DataFrame(wt\_1m, columns = ['Stocks', 'Bonds', 'Gold', 'Real estate'])

port\_ids\_mid = df\_port.loc[(df\_port['returns'] >= 0.05/12) & (df\_port['returns'] <= 0.25/12)].index mid\_ports = wt\_1m.loc[port\_ids\_mid,:].mean(axis=0)

port\_ids\_hi = df\_port.loc[(df\_port['returns'] >= 0.35/12)].index hi\_ports = wt\_1m.loc[port\_ids\_hi,:].mean(axis=0)

port\_ids\_lo = df\_port.loc[(df\_port['returns'] <= -0.05/12)].index lo\_ports = wt\_1m.loc[port\_ids\_lo,:].mean(axis=0)

# Sharpe portfolios

df\_port['sharpe'] = df\_port['returns']/df\_port['risk']\*np.sqrt(12) port\_ids\_sharpe = df\_port[(df\_port['sharpe'] > 0.7)].index sharpe\_ports = wt\_1m.loc[port\_ids\_sharpe,:].mean(axis=0)

# Create graph function

def wt\_graph(ports, title):

fig, ax = plt.subplots(figsize=(12,6)) ax.bar(ports.index.values, ports\*100) for i in range(len(mid\_ports)):

ax.annotate(str(np.round(ports[i],2)\*100), xy=(ports.index.values[i], ports[i]\*100+2), xycoords

ax.set(xlabel = '', ylabel = 'Weigths (%)', title = title, ylim = (0,max(ports)\*100+5)) plt.show()

# Graph weights

wt\_graph(mid\_ports, "Average asset weights for mid-range portfolios") wt\_graph(mid\_ports, "Average asset weights for high return portfolios") wt\_graph(mid\_ports, "Average asset weights for negative return portfolios")

wt\_graph(mid\_ports, "Average asset weights for Sharpe portfolios")

# Built using R 3.6.2

## Load packages suppressPackageStartupMessages({

library(tidyquant) library(tidyverse) library(gtools) library(grid)

})

## Load data

df <- readRDS("port\_const.rds")

dat <- readRDS("port\_const\_long.rds")

sym\_names <- c("stock", "bond", "gold", "realt", "rfr")

## Call simuation functions source("Portfolio\_simulation\_functions.R")

## Prepare sample hist\_avg <- dat %>%

filter(date <= "1991-12-31") %>%

summarise\_at(vars(-date), list(mean = function(x) mean(x, na.rm=TRUE),

sd = function(x) sd(x, na.rm = TRUE))) %>%

gather(key, value) %>%

mutate(key = str\_remove(key, "\_.\*"),

key = factor(key, levels =sym\_names)) %>% mutate(calc = c(rep("mean",5), rep("sd",5))) %>% spread(calc, value)

# Run simulation set.seed(123) sim1 <- list() for(i in 1:1000){

a <- rnorm(60, hist\_avg[1,2], hist\_avg[1,3]) + rnorm(60, 0, hist\_avg[1,3]) b <- rnorm(60, hist\_avg[2,2], hist\_avg[2,3]) + rnorm(60, 0, hist\_avg[2,3]) c <- rnorm(60, hist\_avg[3,2], hist\_avg[3,3]) + rnorm(60, 0, hist\_avg[3,3]) d <- rnorm(60, hist\_avg[4,2], hist\_avg[4,3]) + rnorm(60, 0, hist\_avg[4,3])

df1 <- data.frame(a, b, c, d) cov\_df1 <- cov(df1)

sim1[[i]] <- list(df1, cov\_df1) names(sim1[[i]]) <- c("df", "cov\_df")

}

# Plot random four portfolios ## Sample four return paths

## Note this sampling does not realize in the same way in Rmarkdown/blogdown as in the console. NOt sure

set.seed(123)

samp <- sample(1000,4) graphs <- list()

for(i in 1:8){

if(i <= 4){

graphs[[i]] <- port\_sim(sim1[[samp[i]]]$df,1000,4)

}else{

graphs[[i]] <- port\_sim\_lv(sim1[[samp[i-4]]]$df,1000,4)

}

}

library(grid) gridExtra::grid.arrange(graphs[[1]]$graph +

theme(legend.position = "none") + labs(title = NULL),

graphs[[2]]$graph + theme(legend.position = "none") + labs(title = NULL),

graphs[[3]]$graph + theme(legend.position = "none") + labs(title = NULL),

graphs[[4]]$graph + theme(legend.position = "none") + labs(title = NULL),

ncol=2, nrow=2,

top = textGrob("Four portfolio and return simulations",gp=gpar(fontsize=15)))

# Graph second set gridExtra::grid.arrange(graphs[[5]]$graph +

theme(legend.position = "none") + labs(title = NULL),

graphs[[6]]$graph + theme(legend.position = "none") + labs(title = NULL),

graphs[[7]]$graph + theme(legend.position = "none") + labs(title = NULL),

graphs[[8]]$graph + theme(legend.position = "none") + labs(title = NULL),

ncol=2, nrow=2,

top = textGrob("Four portfolio and return simulations allowing for excluded asse

# Calculate probability of hitting risk-return constraint probs <- c()

for(i in 1:8){

probs[i] <- round(mean(graphs[[i]]$port$returns >= 0.07/12 &

graphs[[i]]$port$risk <=0.1/sqrt(12)),2)\*100

}

## Load data

port\_1m <- readRDS("port\_3m\_sim.rds") ## Graph sample of port\_1m

set.seed(123)

port\_samp = port\_1m[sample(1e6, 1e4),]

port\_samp %>%

mutate(Sharpe = returns/risk) %>% ggplot(aes(risk\*sqrt(12)\*100, returns\*1200, color = Sharpe)) + geom\_point(size = 1.2, alpha = 0.4) +

scale\_color\_gradient(low = "darkgrey", high = "darkblue") + labs(x = "Risk (%)",

y = "Return (%)",

title = "Ten thousand samples from simulation of three million portfolios") + theme(legend.position = c(0.05,0.8), legend.key.size = unit(.5, "cm"),

legend.background = element\_rect(fill = NA))

## Graph histogram port\_1m %>%

mutate(returns = returns\*1200,

risk = risk\*sqrt(12)\*100) %>% gather(key, value) %>% ggplot(aes(value)) +

geom\_histogram(bins=100, fill = 'darkblue') + facet\_wrap(~key, scales = "free",

labeller = as\_labeller(c(returns = "Returns (%)",

risk = "Risk (%)"))) + scale\_y\_continuous(labels = scales::comma) +

labs(x = "",

y = "Count",

title = "Portfolio simulation return and risk histograms")

## Graph quantile returns for total series x\_lim = c("(-15,-5]",

"(-5,5]", "(5,15]",

"(15,25]", "(25,35]")

port\_1m %>%

mutate(returns = cut\_width(returns\*1200, 10)) %>% group\_by(returns) %>%

summarise(risk = median(risk\*sqrt(12)\*100), count = n()/nrow(port\_1m)) %>%

ggplot(aes(returns, count\*100)) + geom\_bar(stat = "identity", fill = "blue") + xlim(x\_lim) +

labs(x = "Return bucket (%)", y = "Frequency (%)",

title = "Frequency of occurrrence for return bucket ")

## Occurrences

mid\_range <- port\_1m %>%

mutate(returns = cut\_width(returns\*1200, 10)) %>% group\_by(returns) %>%

summarise(risk = median(risk\*sqrt(12)\*100), count = n()/nrow(port\_1m)) %>%

filter(as.character(returns) %in% c("(5,15]")) %>% summarise(sum = round(sum(count),2)) %>% as.numeric()\*100

good\_range <- port\_1m %>%

mutate(returns = cut\_width(returns\*1200, 10)) %>% group\_by(returns) %>%

summarise(risk = median(risk\*sqrt(12)\*100), count = n()/nrow(port\_1m)) %>%

filter(as.character(returns) %in% c("(5,15]" , "(15,25]")) %>% summarise(sum = round(sum(count),2)) %>%

as.numeric()\*100

# Set quantiles for graph and labels quants <- port\_1m %>%

mutate(returns = cut(returns\*1200, breaks=c(-Inf, -5, 5, 15, 25, 35, Inf))) %>% group\_by(returns) %>%

summarise(prop = n()/nrow(port\_1m)) %>% select(prop) %>%

mutate(prop = cumsum(prop))

# Calculate quantile

x\_labs <- quantile(port\_1m$returns, probs = unlist(quants))\*1200

x\_labs\_median <- tapply(port\_1m$returns\*1200,

findInterval(port\_1m$returns\*1200, x\_labs), median) %>%

round()

x\_labs\_median\_risk <- tapply(port\_1m$risk\*sqrt(12)\*100, findInterval(port\_1m$risk\*sqrt(12)\*100, x\_labs),

# Graph frequency of occurrence for equal width returns port\_1m %>%

mutate(returns = cut(returns\*1200, breaks=c(-45, -5,5,15,25,35,95))) %>% group\_by(returns) %>%

summarise(risk = median(risk\*sqrt(12)\*100), count = n()/nrow(port\_1m)) %>%

ggplot(aes(returns, count\*100)) + geom\_bar(stat = "identity", fill = "blue") +

geom\_text(aes(returns, count\*100+5, label = paste("Risk: ", round(risk), "%", sep=""))) + geom\_text(aes(returns, count\*100+2,

label = paste("Return: ", x\_labs\_median[-7], "%", sep=""))) + labs(x = "Return bucket (%)",

y = "Frequency (%)",

title = "Frequency of occurrrence for return bucket with median risk and return per bucket")

# High range probability high\_range <- port\_1m %>%

mutate(returns = cut(returns\*1200, breaks=c(-45, -5,5,15,25,35,95))) %>% group\_by(returns) %>%

summarise(risk = median(risk\*sqrt(12)\*100), count = n()/nrow(port\_1m)) %>%

filter(as.character(returns) %in% c("(25,35]", "(35,95]")) %>% summarise(sum = round(sum(count),2)) %>%

as.numeric()\*100

## Identify weights for target portfolios wt\_1m <- readRDS('wt\_3m.rds')

## Portfolio ids

# Mid-range portfolis port\_ids\_mid <- port\_1m %>%

mutate(row\_ids = row\_number()) %>% filter(returns >= 0.05/12, returns < 0.25/12) %>% select(row\_ids) %>%

unlist() %>% as.numeric()

mid\_ports <- colMeans(wt\_1m[port\_ids\_mid,])

# Hi return portfolio port\_ids\_hi <- port\_1m %>%

mutate(row\_ids = row\_number()) %>% filter(returns >= 0.35/12) %>% select(row\_ids) %>%

unlist()

hi\_ports <- colMeans(wt\_1m[port\_ids\_hi,]) # Low return portfolios

port\_ids\_lo <- port\_1m %>% mutate(row\_ids = row\_number()) %>% filter(returns <= -0.05/12) %>% select(row\_ids) %>%

unlist()

lo\_ports <- colMeans(wt\_1m[port\_ids\_lo,]) # Sharpe porfolios

port\_ids\_sharpe <- port\_1m %>% mutate(sharpe = returns/risk\*sqrt(12),

row\_ids = row\_number()) %>% filter(sharpe > 0.7) %>% select(row\_ids) %>%

unlist()

sharpe\_ports <- colMeans(wt\_1m[port\_ids\_sharpe,]) ## Graph portfolio weights

# Function

wt\_graf <- function(assets, weights, title){ data.frame(assets = factor(assets, levels = assets),

weights = weights) %>% ggplot(aes(assets, weights\*100)) + geom\_bar(stat = "identity", fill="blue") +

geom\_text(aes(assets ,weights\*100+3, label = round(weights,2)\*100)) + labs(x='',

y = "Weights (%)", title = title)

}

assets = c("Stocks", "Bonds", "Gold", "Real Estate") # Graph diferent weights

wt\_graf(assets, mid\_ports, "Average asset weights for mid-range portfolios")

wt\_graf(assets, hi\_ports, "Average asset weights for high return portfolios") wt\_graf(assets, lo\_ports, "Average asset weights for negative return portfolios") wt\_graf(assets, sharpe\_ports, "Average asset weights for Sharpe constraints")