

```

marketPortfolio = function(merged,
                           rf,
                           returnNames,
                           weightNames,
                           graph=FALSE,
                           points=500,
                           maxWeight=.334,
                           Debug=FALSE) {

  #create an empty data frame for the portfolio weights
  weights = data.frame(t(rep(NA,length(weightNames))))
  colnames(weights) = weightNames
  weights = weights[-1,]

  #Calculate Annualized Returns
  t = table.AnnualizedReturns(merged[,returnNames])

  #Range to optimize over
  maxRet = max(t['Annualized Return',]) - .005
  minRet = min(t['Annualized Return',]) + .005

  #set the max and min for the returns. Cap the upper
  #limit to 50% and the lower to .005%
  maxRet = min(.5,maxRet)
  minRet = max(0.005,minRet)

  #if all series have negative expected return, then
  #only find the portfolio with the max return
  if (maxRet < 0) {
    minRet = maxRet
    points = 1
  }

  #Debugging Print
  if (Debug) {
    print("Max Return")
    print(maxRet)
    print("Min Return")
    print(minRet)
  }

  #portfolio.optim cannot have NA values in the time series, filter
  #them out
  m2 = removeNA(merged[,returnNames])

  er = NULL
  eStd = NULL

  #loop through finding the optimum portfolio for return levels between
  #the range found above
  #
  #portfolio.optim uses daily returns, so we have to adjust accordingly
  ok = FALSE
  for (i in seq(minRet,maxRet,length.out=points)) {
    pm = 1+i
    pm = log(pm)/255
    #Debugging Print

```

```

    if (Debug) {
      print("Finding Optimum for")
      print("ER")
      print(pm)
      print(exp(pm*255)-1)
    }
    #optimization. limit weights to <= maxWeight
    #surround in tryCatch to catch unobtainable values
    opt = tryCatch(portfolio.optim(m2,
      pm=pm,

reshigh=rep(maxWeight,length(weightNames)),
      shorts = FALSE),
      error=function(err) return(NULL))

#Debug print of opt results
if (Debug) print(opt)

#check to see if feasible solution exists
if (!is.null(opt)){
  er = c(er,exp(pm*255)-1)
  eStd = c(eStd,opt$ps*sqrt(255))
  w = t(opt$pw)
  colnames(w) = weightNames
  weights = rbind(weights,w)

  #update the OK variable
  ok = (ok | TRUE)
} else {
  print("ERROR IN THIS TRY")
  #update the OK variable
  ok = (ok | FALSE)
}
}

#if no feasible solutions were found for the frontier,
#return NULL
if (!ok){
  return (NULL)
}

solution = weights
solution$er = er
solution$eStd = eStd

#find the index values for the minimum Std and the max Er
minIdx = which(solution$eStd == min(solution$eStd))
maxIdx = which(solution$er == max(solution$er))

if (Debug){
  print(minIdx)
  print(maxIdx)
}

#subset the results
subset = solution[minIdx:maxIdx,c("er","eStd")]

```

```

subset$nAbove = NA

#for each value in the subset, count the number of points
#that lay below a line drawn through the point and the
#RF asset
for (i in seq(1,maxIdx-minIdx+1)){
  toFit = data.frame(er=rf,eStd=0)
  toFit = rbind(toFit,subset[i,c("er","eStd")])
  fit = lm(toFit$er ~ toFit$eStd)
  poly = polynomial(coef = fit$coefficients)
  toPred = subset
  colnames(toPred) = c("actEr","eStd")
  toPred$er = predict(poly,toPred[, "eStd"])
  toPred$diff = toPred$er - toPred$actEr
  subset[i,"nAbove"] = nrow(toPred[which(toPred$diff > 0),])
}

#get the point of tangency -- where the number of points
#below the line is maximized
max = max(subset$nAbove)
er = subset[which(subset$nAbove == max),"er"]
eStd = subset[which(subset$nAbove == max),"eStd"]

#if more than one portfolio is found, return the first
if (length(er) > 1){
  er = er[1]
  eStd = eStd[1]
}

#index of the market portfolio
idx = which(solution$er == er & solution$eStd == eStd)

if (Debug){
  print("solution")
  print(er)
  print(eStd)
  print(solution[idx,])
}

#Draw the line if requested
if (graph){
  maxStd = max(solution$eStd) + .02
  maxRetg = max(solution$er) + .02
  plot(solution$eStd,
        solution$er,
        xlim=c(0,maxStd),
        ylim=c(0,maxRetg),
        ylab="Expected Yearly Return",
        xlab="Expected Yearly Std Dev",
        main="Efficient Frontier",
        col="red",
        type="l",
        lwd=2)
  abline(v=c(0), col="black", lty="dotted")
  abline(h=c(0), col="black", lty="dotted")

  toFit = data.frame(er=rf,eStd=0)

```

```
    toFit = rbind(toFit,solution[idx,c("er","eStd")])
    fit = lm(toFit$er ~ toFit$eStd)
    abline(coef=fit$coefficients,col="blue",lwd=2)
  }

  #Return the market portfolio wieghts and eStd and eR
  out = solution[idx,]
  return (out)
}
```