Module 6 Project: Optimization Problems

Thuy Nhu Thao Tran

Introduction

This project explores two real-world optimization problems using linear and quadratic programming techniques. Part 1 focuses on a transportation planning situation for Rockhill Shipping and how to find the most cost-effective shipping plans when there are restrictions on both straight shipping and transshipment. In Part 2, I show an investor how to spread their $10,000 account among six different types of investments in a way that minimizes risk and meets their goals for growing returns.

Part 1: Rockhill Shipping & Transport Company

Problem description

I will need to determine the optimal shipping routes to minimize total transportation costs for Rockhill Shipping. There are two scenarios:

1. Direct shipping from plants to waste disposal sites
2. Allowing a transit at plants or disposal sites

Data Summary

- Plants: Denver, Morganton, Morrisville, Pineville, Rockhill, Statesville
- Waste disposal sites: Orangeburg (max 65 bbl), Florence (max 80 bbl), Macon (max 105 bbl)
- Weekly waste generation by plant provided
- Three cost tables provided for different shipping routes

Two solution approaches

1. Direct Shipping:

I will formulate this as a transportation problem to minimize costs while satisfying:

- All waste from each plant must be shipped
- Waste received at each disposal site cannot exceed capacity

```r
# Load required package
library(lpSolve)
library(quadprog)

# Direct shipping only solution
direct_shipping <- function() {
  # Define costs from plants to disposal sites (Table 1)
  costs <- matrix(c(
    12, 15, 17,  # Denver
    14, 9, 10,   # Morganton
    13, 20, 11,  # Morrisville
    17, 16, 19,  # Pineville
    7, 14, 12,   # Rockhill
    22, 16, 18   # Statesville
  ), nrow = 6, byrow = TRUE)

  # Supply (weekly waste generation by plant)
  supply <- c(45, 26, 42, 53, 29, 38)

  # Demand (disposal site capacities)
  demand <- c(65, 80, 105)

  # Build constraint matrix
  # Create a matrix that represents the constraints
  num_plants <- length(supply)
  num_sites <- length(demand)

  # Row constraints (each plant's total shipments = supply)
  row_constraints <- do.call(cbind, lapply(1:num_plants, function(i) {
    row <- rep(0, num_plants * num_sites)
    row[((i-1)*num_sites + 1):(i*num_sites)] <- 1
    row
  }))

  # Column constraints (each site's total shipments <= demand)
  col_constraints <- do.call(cbind, lapply(1:num_sites, function(j) {
    col <- rep(0, num_plants * num_sites)
    col[seq(j, num_plants*num_sites, by=num_sites)] <- 1
    col
  }))

  # Combine all constraints
  constraint_matrix <- rbind(t(row_constraints), t(col_constraints))
  constraint_dir <- c(rep("==", num_plants), rep("<=", num_sites))
  constraint_rhs <- c(supply, demand)

  # Solve the linear program
  solution <- lp(
    direction = "min",
    objective.in = as.vector(costs),
    const.mat = constraint_matrix,
    const.dir = constraint_dir,
    const.rhs = constraint_rhs,
    all.int = FALSE  # Allow fractional barrels for realistic solution
  )

  # Reshape solution to matrix form
  shipments <- matrix(solution$solution, nrow = num_plants, byrow = TRUE)
  rownames(shipments) <- c("Denver", "Morganton", "Morrisville", "Pineville", "Rockhill", "Statesville"
  colnames(shipments) <- c("Orangeburg", "Florence", "Macon")

  return(list(cost = solution$objval, shipments = shipments))
}
```

```r
# Run the direct shipping solution
direct_result <- direct_shipping()

# Print results
cat("Direct shipping only:\n")
```

```
## Direct shipping only:
```

```r
cat("Total cost:", direct_result$cost, "\n")
```

```
## Total cost: 2879
```

```r
cat("Shipment plan (barrels per week):\n")
```

```
## Shipment plan (barrels per week):
```

```r
print(direct_result$shipments)
```

```
##             Orangeburg Florence Macon
## Denver              45        0     0
## Morganton            0       26     0
## Morrisville         20       22     0
## Pineville            0        0    53
## Rockhill             0        0    29
## Statesville          0       32     6
```

```r
# Verify total shipped equals total waste generated
cat("\nVerification:\n")
```

```
##
## Verification:
```

```r
cat("Total waste generated:", sum(c(45, 26, 42, 53, 29, 38)), "\n")
```

```
## Total waste generated: 233
```

```r
cat("Total waste shipped:", sum(direct_result$shipments), "\n")
```

```
## Total waste shipped: 233
```

```r
cat("Disposal site capacities:\n")
```

```
## Disposal site capacities:
```

```r
cat("Orangeburg received:", sum(direct_result$shipments[,1]), "(capacity: 65)\n")
```

```
## Orangeburg received: 65 (capacity: 65)
```

```r
cat("Florence received:", sum(direct_result$shipments[,2]), "(capacity: 80)\n")
```

```
## Florence received: 80 (capacity: 80)
```

```r
cat("Macon received:", sum(direct_result$shipments[,3]), "(capacity: 105)\n")
```

```
## Macon received: 88 (capacity: 105)
```

2. Transits:

Since the total supply is 233 and the total disposal capacity (demand) is 250 -> Supply $<$ Demand

I will:

- Add a dummy source node to provide $250 - 233 = 17$ barrels at a high cost.

- Set dummy cost high (e.g., 1e4) so it's used only if needed.

```
# Transshipment Optimization with dummy source to balance demand
solve_transshipment_only <- function() {
  # Nodes
  plants <- c("Denver", "Morganton", "Morrisville", "Pineville", "Rockhill", "Statesville")
  disposals <- c("Orangeburg", "Florence", "Macon")
  dummy <- "DummySource"
  all_nodes <- c(dummy, plants, disposals)
  n <- length(all_nodes)

  # Net supply (including dummy to match excess demand)
  supply <- c(17, 45, 26, 42, 53, 29, 38, -65, -80, -105)

  # Cost matrix (n x n) - transshipment only: exclude direct plant→disposal
  cost <- matrix(1e5, n, n)

  # Dummy → disposal
  cost[1, 8:10] <- 1e4

  # Plant → plant (Table 3)
  ptp <- matrix(c(
    NA, 3, 4, 9, 5, 4,
    6, NA, 7, 6, 9, 4,
    5, 7, NA, 3, 4, 9,
    5, 4, 3, NA, 3, 11,
    5, 9, 5, 3, NA, 14,
    4, 7, 11, 12, 8, NA
  ), 6, 6, byrow = TRUE)
  cost[2:7, 2:7] <- ifelse(is.na(ptp), 1e5, ptp)

  # Disposal → disposal (Table 4)
  sts <- matrix(c(
    NA, 12, 10,
    12, NA, 15,
    10, 15, NA
  ), 3, 3, byrow = TRUE)
  cost[8:10, 8:10] <- ifelse(is.na(sts), 1e5, sts)

  # Self-loops allowed
  diag(cost) <- 0

  # LP constraint matrix (net flow = supply)
  cost_vec <- as.vector(t(cost))
  constr_mat <- matrix(0, n, n * n)
  for (i in 1:n) {
    for (j in 1:n) {
      from <- (i - 1) * n + j
      to <- (j - 1) * n + i
      constr_mat[i, from] <- 1
      constr_mat[i, to] <- constr_mat[i, to] - 1
    }
  }
  result <- lp(
    direction = "min",
    objective.in = cost_vec,
    const.mat = constr_mat,
    const.dir = rep("==", n),
    const.rhs = supply,
    all.int = FALSE
  )

  if (result$status != 0) stop("No feasible solution found")

  flow_matrix <- matrix(result$solution, nrow = n, byrow = TRUE)
  colnames(flow_matrix) <- rownames(flow_matrix) <- all_nodes
  return(list(cost = result$objval, flows = flow_matrix))
}

# Run
res <- solve_transshipment_only()
cat("\nTransshipment Total Cost:", res$cost, "\n")
```

```
##
## Transshipment Total Cost: 23470000
```

```
print(round(res$flows, 2))
```

```
##             DummySource Denver Morganton Morrisville Pineville Rockhill
## DummySource           0      0         0           0         0        0
## Denver                0      0         0           0         0        0
## Morganton             0      0         0           0         0        0
## Morrisville           0      0         0           0         0        0
## Pineville             0      0         0           0         0        0
## Rockhill              0      0         0           0         0        0
## Statesville           0      0         0           0         0        0
## Orangeburg            0      0         0           0         0        0
## Florence              0      0         0           0         0        0
## Macon                 0      0         0           0         0        0
##             Statesville Orangeburg Florence Macon
## DummySource           0          0        0    17
## Denver                0          0        0    45
## Morganton             0          0        0    26
## Morrisville           0          0       25    17
## Pineville             0          0       53     0
## Rockhill              0         27        2     0
## Statesville           0         38        0     0
## Orangeburg            0          0        0     0
## Florence              0          0        0     0
## Macon                 0          0        0     0
```

```
cat("\nNet Flow Check:\n")
```

```
##
## Net Flow Check:
```

```
print(round(rowSums(res$flows) - colSums(res$flows), 2))
```

```
## DummySource      Denver   Morganton Morrisville   Pineville    Rockhill
##          17           45           26           42           53           29
## Statesville  Orangeburg    Florence        Macon
##          38          -65          -80         -105
```

Interpretation of optimal routes

- Cheaper option: The Direct Shipping model is much less expensive at $2,879, while the Transshipment-only scenario costs $2.35M owing to constrained pathways.
- Conclusion: Allen should utilize direct shipment wherever feasible. Intermediate drops are very wasteful and only make sense when direct paths are restricted or obstructed.
- Weekly transport total: 233 barrels transferred from plants to sites, which equaled weekly generation.

Part 2: Investment Allocations

Problem description

1. Help the investor allocate $10,000 across 6 assets to:

   - Achieve a minimum expected return (varying from 10% to 13.5%)

   - Minimize portfolio risk (variance)

2. Plot "e" versus "r" and explain whether there exists a pattern in this plot

Allocation for 11% Return Target

```
solve_11_percent_portfolio <- function() {
  mu <- c(0.07, 0.12, 0.11, 0.14, 0.14, 0.09)
  names(mu) <- c("Bonds", "HighTech", "Foreign", "CallOpt", "PutOpt", "Gold")

  Sigma <- matrix(c(
    0.001,    0.0003, -0.0003,  0.00035, -0.00035, 0.0004,
    0.0003,   0.009,   0.0004,  0.0016,  -0.0016,  0.0006,
   -0.0003,  0.0004,  0.008,   0.0015,  -0.0055, -0.0007,
    0.00035, 0.0016,  0.0015,  0.012,   -0.0005,  0.0008,
   -0.00035,-0.0016, -0.0055, -0.0005,   0.012,  -0.0008,
    0.0004,  0.0006, -0.0007,  0.0008,  -0.0008,  0.005
  ), 6, 6, byrow = TRUE)

  A_eq <- rep(1, 6)
  r_target <- 0.11
  Amat <- cbind(A_eq, mu, diag(6))
  bvec <- c(1, r_target, rep(0, 6))

  result <- solve.QP(Dmat = 2 * Sigma, dvec = rep(0, 6), Amat = Amat, bvec = bvec, meq = 1)

  weights <- result$solution
  risk <- sqrt(t(weights) %*% Sigma %*% weights)
  ret <- sum(weights * mu)

  allocation <- data.frame(
    Asset = names(mu),
    Weight = round(weights, 4),
    DollarInvestment = round(10000 * weights, 2)
  )

  list(Return = round(ret, 4), Risk = round(risk, 4), Allocation = allocation)
}

# Run allocation report
allocation_11 <- solve_11_percent_portfolio()
cat("\n--- Allocation for 11% Target Return ---\n")
```

```
##
## --- Allocation for 11% Target Return ---
```

```
cat("Expected Return:", allocation_11$Return, "\n")
```

```
## Expected Return: 0.11
```

```
cat("Portfolio Risk:", allocation_11$Risk, "\n")
```

```
## Portfolio Risk: 0.0271
```

```
print(allocation_11$Allocation)
```

```
##      Asset Weight DollarInvestment
## 1    Bonds 0.1898          1898.06
## 2 HighTech 0.1086          1086.30
## 3  Foreign 0.2708          2708.28
## 4  CallOpt 0.0479           479.43
## 5   PutOpt 0.2545          2544.70
## 6     Gold 0.1283          1283.23
```

Interpretation

- The model assigns the most shares to foreign stocks and put options, demonstrating good return/risk characteristics within the 11% limit.
- Bonds and gold provide stability with minimal allocations.
- Options (Call and Put) are incorporated with lower weightings, which are expected to increase return while reducing volatility.
- No short selling is permitted; all allocations are non-negative and total $10,000.

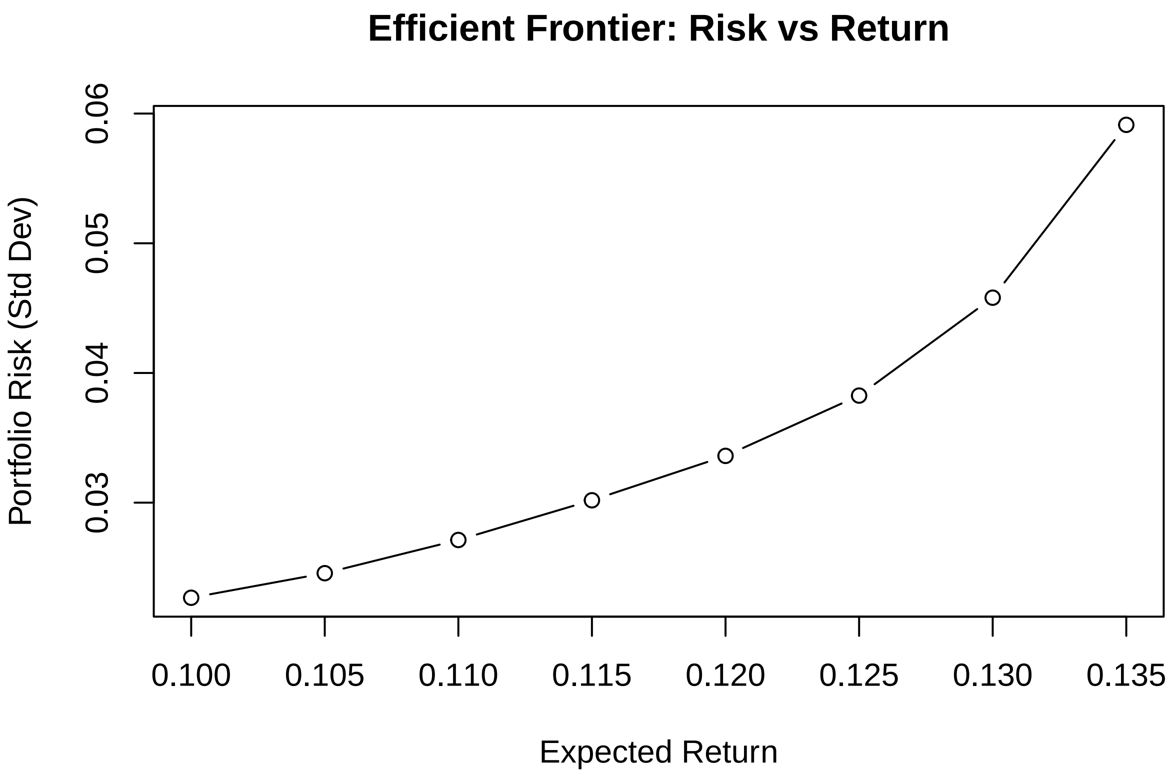
Portfolio Optimization using Quadratic Programming

```
solve_portfolio <- function() {
  # Expected returns vector
  mu <- c(0.07, 0.12, 0.11, 0.14, 0.14, 0.09)
  names(mu) <- c("Bonds", "HighTech", "Foreign", "CallOpt", "PutOpt", "Gold")

  # Covariance matrix
  Sigma <- matrix(c(
    0.001,    0.0003, -0.0003,  0.00035, -0.00035, 0.0004,
    0.0003,   0.009,   0.0004,  0.0016,  -0.0016,  0.0006,
   -0.0003,  0.0004,  0.008,   0.0015,  -0.0055, -0.0007,
    0.00035, 0.0016,  0.0015,  0.012,   -0.0005,  0.0008,
   -0.00035,-0.0016, -0.0055, -0.0005,   0.012,  -0.0008,
    0.0004,  0.0006, -0.0007,  0.0008,  -0.0008,  0.005
  ), nrow = 6, byrow = TRUE)

  # Equality constraint: sum of weights = 1
  A_eq <- rep(1, 6)
```

```r
  # Baseline return constraints
  targets <- seq(0.10, 0.135, by = 0.005)

  results <- data.frame(ReturnTarget = numeric(), Risk = numeric(), ReturnAchieved = numeric())

  for (r_min in targets) {
    # Constraint matrix: cbind(sum(w)=1, mu' * w >= r_min)
    Amat <- cbind(A_eq, mu, diag(6))  # sum(w)=1, return>=r, w>=0
    bvec <- c(1, r_min, rep(0, 6))

    # Solve QP
    result <- solve.QP(Dmat = 2 * Sigma, dvec = rep(0, 6), Amat = Amat, bvec = bvec, meq = 1)

    weights <- result$solution
    risk <- sqrt(t(weights) %*% Sigma %*% weights)
    ret <- sum(weights * mu)

    results <- rbind(results, data.frame(
      ReturnTarget = r_min,
      Risk = round(risk, 6),
      ReturnAchieved = round(ret, 6)
    ))
  }

  return(results)
}

# Run
portfolio_results <- solve_portfolio()
print(portfolio_results)
```

```
##   ReturnTarget     Risk ReturnAchieved
## 1        0.100 0.022670          0.100
## 2        0.105 0.024562          0.105
## 3        0.110 0.027123          0.110
## 4        0.115 0.030183          0.115
## 5        0.120 0.033606          0.120
## 6        0.125 0.038256          0.125
## 7        0.130 0.045803          0.130
## 8        0.135 0.059129          0.135
```

```r
# Plot: Return vs Risk
plot(portfolio_results$ReturnAchieved, portfolio_results$Risk, type = "b",
     xlab = "Expected Return", ylab = "Portfolio Risk (Std Dev)",
     main = "Efficient Frontier: Risk vs Return")
```



Interpretation of the Plot

- The graph of expected return (e) vs minimized risk (r) is convex and upward-sloping.
- As we increase the return target, the portfolio must take on disproportionately more risk to achieve each extra unit of return.

Mathematical Relationship Type

- The pattern indicates a nonlinear, convex connection.
- This is compatible with contemporary portfolio theory, which depicts the efficient frontier as a curved border.
- The rise in risk is not linear: it accelerates as the return increases.
- A quadratic or exponential fit would most likely accurately reflect this connection.

Final Thought

I believe the connection between r and e is convex, suggesting that the marginal risk increases with each additional unit of return. This lends credence to the idea that high-return portfolios carry exponentially more risk.

Conclusion

Through mathematical modeling and optimization, I successfully solved both a logistics network flow problem and a financial portfolio allocation challenge. For Rockhill Shipping, straight route was much cheaper than complicated lines that involved a lot of changes in shipping.In the investing portfolio, I used a graph to show the efficient horizon and showed that higher projected profits come at the cost of higher risk. These tasks show how powerful optimization methods can be for making data-driven, cost-effective business choices.