CFRM507 Project 1 - Final Report

Tsing Ouyang
CFRM 507: Optimization Methods in Finance
Student Number: 2528985
j22ouyan@uw.edu

2025-10-28

Optimal Solution and Recommendation

The optimal investment strategy derived from solving the linear programming model. The solution minimizes the total economic cost while satisfying all benefit payments and portfolio constraints.

1. Investment Cost

The plan sponsor must make a minimum initial investment to fully fund the pension's obligations.

- **Total Economic Cost (Z):** `[$9,310,627,859.35]`
- **Total Initial Outlay:** `[``$9,310,627,859.35``]`
  - **Initial Bond Purchases:** `[``$9,310,627,859.35``]`
  - **Initial Cash Deposit (s_0):** `[``$0``]`
- **Residual Cash (s_{80}):** `[$0]`

The plan sponsor needs to invest $\$9,310,627,859.35$, and among all of initial investment, all of the $\$9,310,627,859.35$ goes to bond and 0 goes to the cash account. Also, the total economic cost includes the initial cash deposit and leftover cash at Quarter 80, where both are zero $s_0 = s_{80} = 0$ and the minimized objective function (optimal value) is $\$9,310,627,859.35$

2. Recommended Securities (Purchased Bond)

The optimal portfolio allocation is detailed in Appendix. The strategy primarily uses a mix of bonds to match cash flows. The table below shows all recommended purchases with a dollar value greater than $1.00, sorted by investment size.

Table: Top 20 Optimal Bond Purchases

Bond_ID	Type	Units_Purchased	Dollar_Value	Percent_of_Bond_Portfolio
78	3	992,517.3	1,000,000,000	0.10740414
39	2	465,410.1	560,116,358	0.06015882
77	3	487,760.5	484,407,112	0.05202733

Bond_ID	Type	Units_Purchased	Dollar_Value	Percent_of_Bond_Portfolio
40	2	401,745.4	470,588,525	0.05054316
32	2	256,994.1	405,169,232	0.04351685
28	2	371,901.6	403,141,298	0.04329905
41	2	354,737.2	401,307,099	0.04310204
29	2	311,403.5	400,103,651	0.04297279
30	2	361,877.5	398,481,454	0.04279856
24	2	354,645.8	393,862,479	0.04230246
34	2	362,369.1	391,938,418	0.04209581
26	2	249,184.6	391,431,680	0.04204138
36	2	343,544.3	361,068,473	0.03878025
35	2	278,599.8	342,680,506	0.03680531
50	2	314,536.4	321,371,260	0.03451660
37	2	265,861.8	320,105,557	0.03438066
38	2	302,440.3	312,756,557	0.03359135
44	2	272,049.9	287,877,748	0.03091926
45	2	228,649.1	241,586,085	0.02594735
82	3	237,116.4	237,718,460	0.02553195

3. Bonds Allocation

This portfolio satisfies all defined constraints and the optimal portfolio allocation shows below:

- **Dollar-Weighted Avg. Credit Score:** `[0.0354]` (Limit: 0.4)
- **Allocation CIPS:** `[1.50%]` (Limit: 30%)
- **Allocation TIPS:** `[80.00%]` (Limit: 80%)
- **Allocation Nominal:** `[18.50%]` (Limit: 80%)
- **Allocation SMAs:** `[0.00%]` (Limit: 10%)
- **Nominal CF Ratio:** `[20.00%]` (Limit: 20%)

The primary remaining risk is the inflation risk, which we have modeled by deflating nominal cash flows at an assumed 3% over the cash-like investment accures at 2%. the inflation exceeds this cash rate, the purchasing power of the nominal cash flows will be less than anticipated.

4. Shadow Prices (Sensitivity Analysis)

Below are the shadow prices of each constraint, which indicates how much the total cost would decrease if for every extra dollar of relative constraints could make, since the rearrangement of the constraint matrix. The RHS of the constraint have different meanings. In summary, for these shadow prices are 0, they are non-blinding constraints (not affecting the total cost by small relaxation). For the shadow prices are negative, the plan sponsor is forced to choose a more costly bond by the constraint.

- **Credit Quality Limit:** `[0]`
- **Allocation CIPS Limit:** `[0]`
- **Allocation TIPS Limit:** `[-1.25]`

- **Allocation Nominal Bonds Limit:** `[0]`
- **Allocation SMAs Limit:** `[0]`
- **Nominal CF Ratio Limit:** `[-1.2115586]`

That is, for every additional dollar invested in TIPS, the total cost will be dropped by 1.25; similarly, for every additional dollar invested in TIPS, the total cost will be dropped by 1.2115586 in intuitive way.

5. Sensitivity Analysis (Inflation Rate and Cash Accrues Rate)

By one basis point relaxation of the rates, the total cost changed from $\$9,310,627,859.35$ to

- **Inflation Rate (3% - 3.01%):** `[$9,311,011,305.15]`
- **Cash Accrues Rate (2% - 2.01%):** `[$9,310,401,151.72]`

That is, the one basis point of inflation rate would cause the cost increased by 1 million dollars and the one basis point of cash rate will have the cost decreased by around 227 thousand dollars. The primary remaining risk is the inflation risk, which the inflation exceeds this cash rate, the purchasing power of the nominal cash flows will be less than anticipated. And the cost will be impacted heavily by the inflation rate.

Appendix

```r
# --- 1. Load Libraries ---
# install.packages("Rglpk") # Uncomment this line if you haven't installed Rglpk
library(Rglpk)
```

```
## Loading required package: slam

## Using the GLPK callable library version 5.0
```

```r
# --- 2. Load Data Files ---

# Load Benefit Payments
benefit_data <- read.table("BenefitPayments.txt", header = FALSE)
n_quarters <- benefit_data[1, 1] # Should be 80
# Vector of benefit payments (B_t) for t=1 to 80
B <- benefit_data[-1, 1]

# Load Bond Data
bond_data_raw <- read.table("BondData.txt", header = FALSE, skip = 1)
n_bonds <- nrow(bond_data_raw) # Should be 230
colnames(bond_data_raw) <- c("Type", "Rating", "Price", "UpperBound")

# Extract bond data into vectors
P <- bond_data_raw$Price         # Price (P_i)
S <- bond_data_raw$Rating        # Credit Score (S_i)
U <- bond_data_raw$UpperBound      # Upper $ Bound (U_i)
Type <- bond_data_raw$Type        # Type (1=CIPS, 2=TIPS, 3=Nominal Bonds, 4=SMA)

# Load Cash Flow Matrices (Rows = Quarters, Cols = Bonds)
# Nominal Cash Flows (N_it)
N <- as.matrix(read.table("NomCashflow.txt", header = FALSE))
# Real (Inflation-Linked) Cash Flows (R_it)
R <- as.matrix(read.table("RealCashflow.txt", header = FALSE))

# --- 3. Assign Parameters ---

saving_rate <- 0.0201
inflation_rate <- 0.03
credit_limit <- 0.40
allocation_limit_CIPS <- 0.30
allocation_limit_TIPS <- 0.80
allocation_limit_Nominal_Bond <- 0.80
allocation_limit_SMAs <- 0.10
nominal_cash_Limit <- 0.20

# --- 4. Define Decision Variables ---

n_vars <- n_bonds + n_quarters + 1 # 230 (x_i) + 81 (c_t, t=0 to 80)
# Vars 1 to 230 are bond units x_i
# Vars 231 to 311 are cash c_0, c_1, ..., c_80
# --- 5. Define Objective Function ---
# Minimize Z = sum(x_i * P_i) + c_0 + c_80

obj <- numeric(n_vars)
# Bond purchase cost
obj[1:n_bonds] <- P
# Initial cash deposit c_0
obj[n_bonds + 1] <- 1
# Final cash holding c_80
obj[n_vars] <- 1

# --- 6. Define Constraints (mat, dir, rhs) ---

# Initialize constraint matrix, direction, and RHS vectors
mat_list <- list()
dir_vec <- c()
rhs_vec <- c()

# --- Constraint 1: Benefit Payment Coverage (80 constraints) ---
# (c_{t-1} * e^((saving_rate-inflation_rate/4)) + sum(x_i * (R_it + N_it / e^(-inflation_limit*t/4)) -
for (t in 1:n_quarters) {
  row <- numeric(n_vars)

  # sum(x_i * (R_it + N_it * e^(-inflation_limit*t/4)))
  inflation_discount <- exp(inflation_rate*t/4)
  row[1:n_bonds] <- R[t, ] + (N[t, ] / inflation_discount)

  # + c_{t-1} * e^((saving_rate-inflation_rate/4))
  # Note: c_0 is var 231, c_1 is 231, ..., c_{t-1} is 231 + (t-1)
  row[n_bonds + t] <- exp((saving_rate-inflation_rate)/4)

  # - c_t
  row[n_bonds + t + 1] <- -1

  mat_list[[length(mat_list) + 1]] <- row
  dir_vec <- c(dir_vec, ">=")
  rhs_vec <- c(rhs_vec, B[t])
}

# --- Constraint 2: Credit Quality (1 constraint) ---
# sum(x_i * (P_i * S_i - credit_limit * P_i)) <= 0
row <- numeric(n_vars)
row[1:n_bonds] <- P * S - credit_limit * P

mat_list[[length(mat_list) + 1]] <- row
dir_vec <- c(dir_vec, "<=")
rhs_vec <- c(rhs_vec, 0)

# --- Constraint 3: Maximum Allocation by Bond Type (4 constraints) ---
# CIPS (Type 1) <= 30%
row_cips <- numeric(n_vars)
row_cips[1:n_bonds] <- ifelse(Type == 1, P * (1 - allocation_limit_CIPS), P * (-allocation_limit_CIPS))
mat_list[[length(mat_list) + 1]] <- row_cips
```

```r
dir_vec <- c(dir_vec, "<=")
rhs_vec <- c(rhs_vec, 0)

# TIPS (Type 2) <= 80%
row_tips <- numeric(n_vars)
row_tips[1:n_bonds] <- ifelse(Type == 2, P * (1 - allocation_limit_TIPS), P * (-allocation_limit_TIPS))
mat_list[[length(mat_list) + 1]] <- row_tips
dir_vec <- c(dir_vec, "<=")
rhs_vec <- c(rhs_vec, 0)

# Nominal (Type 3) <= 80%
row_nom <- numeric(n_vars)
row_nom[1:n_bonds] <- ifelse(Type == 3, P * (1 - allocation_limit_Nominal_Bond), P * (-allocation_limit
mat_list[[length(mat_list) + 1]] <- row_nom
dir_vec <- c(dir_vec, "<=")
rhs_vec <- c(rhs_vec, 0)

# SMAs (Type 4) <= 10%
row_smas <- numeric(n_vars)
row_smas[1:n_bonds] <- ifelse(Type == 4, P * (1 - allocation_limit_SMAs), P * (-allocation_limit_SMAs))
mat_list[[length(mat_list) + 1]] <- row_smas
dir_vec <- c(dir_vec, "<=")
rhs_vec <- c(rhs_vec, 0)

# --- Constraint 4: Nominal Cash Flow Limit (1 constraint) ---
# sum(x_i * [0.8 * sum_t(N_it) - 0.2 * sum_t(R_it)]) <= 0
Total_N <- colSums(N) # Total nominal CF per bond
Total_R <- colSums(R) # Total real CF per bond

row <- numeric(n_vars)
row[1:n_bonds] <- (1-nominal_cash_Limit) * Total_N - (nominal_cash_Limit * Total_R)

mat_list[[length(mat_list) + 1]] <- row
dir_vec <- c(dir_vec, "<=")
rhs_vec <- c(rhs_vec, 0)

# --- Combine constraints into a single matrix ---
mat_constraints <- do.call(rbind, mat_list)

# --- 6. Define Bounds ---
# Constraint 5: Individual Bond Availability (x_i * P_i <= U_i) => x_i <= U_i / P_i
# Constraint 6: Non-Negativity (x_i >= 0, c_t >= 0)

# Rglpk handles division by zero gracefully (Inf), but we'll be careful
upper_x <- U / P
# Handle any P_i = 0 cases, though unlikely for price
upper_x[is.infinite(upper_x)] <- 1e20 # A very large number
upper_x[is.na(upper_x)] <- 0       # If 0/0, can't buy

bounds <- list(
  # Lower bounds: x_i >= 0, c_t >= 0
  lower = list(ind = 1:n_vars, val = rep(0, n_vars)),
  # Upper bounds: x_i <= U_i / P_i. Cash (c_t) is unbounded above.
  upper = list(ind = 1:n_bonds, val = upper_x)
)

# --- 7. Solve the LP ---
solution <- Rglpk_solve_LP(
  obj = obj,
  mat = mat_constraints,
  dir = dir_vec,
  rhs = rhs_vec,
  bounds = bounds,
  max = FALSE # We are minimizing
)

# --- 8. Display Solution ---

if (solution$status == 0) {
  # Extract solution variables
  vars <- solution$solution
  x <- vars[1:n_bonds]
  c_cash <- vars[(n_bonds + 1):n_vars] # c_0, c_1, ..., c_80

  total_cost_Z <- solution$optimum
  initial_bond_cost <- sum(x * P)
  initial_cash_c0 <- c_cash[1] # c_0
  final_cash_c80 <- c_cash[n_quarters + 1] # c_80

  purchases <- data.frame(
    Bond_ID = 1:n_bonds,
    Type = Type,
    Units_Purchased = x,
    Dollar_Value = x * P,
    Percent_of_Bond_Portfolio = (x * P) / initial_bond_cost
  )

  # Filter for non-trivial purchases
  purchases_filtered <- purchases[purchases$Dollar_Value > 1, ]
  purchases_filtered <- purchases_filtered[order(-purchases_filtered$Dollar_Value), ]

  if (nrow(purchases_filtered) > 0) {
    print(head(purchases_filtered, 20), row.names = FALSE)
  } else {
    cat("No significant bond purchases recommended.\n")
  }


  # Credit Quality
  avg_credit <- sum(x * P * S) / initial_bond_cost

  # Allocation by Type
  alloc_cips <- sum(purchases$Dollar_Value[purchases$Type == 1]) / initial_bond_cost
  alloc_tips <- sum(purchases$Dollar_Value[purchases$Type == 2]) / initial_bond_cost
  alloc_nom <- sum(purchases$Dollar_Value[purchases$Type == 3]) / initial_bond_cost
  alloc_smas <- sum(purchases$Dollar_Value[purchases$Type == 4]) / initial_bond_cost
  # Nominal CF Limit
  total_nom_cf <- sum(Total_N * x)
  total_real_cf <- sum(Total_R * x)
  nom_cf_ratio <- total_nom_cf / (total_nom_cf + total_real_cf)



  constraint_labels <- c(
    paste("Benefit_Q", 1:n_quarters, sep = ""), # Constraints 1-80
    "Credit_Quality_Limit",                     # Constraint 81
    "Alloc_CIPS_Limit",                         # Constraint 82
    "Alloc_TIPS_Limit",                         # Constraint 83
    "Alloc_Nominal_Limit",                      # Constraint 84
    "Alloc_SMAs_Limit",                         # Constraint 85
    "Nominal_CF_Limit"                          # Constraint 86
  )

  duals_df <- data.frame(
    Constraint_ID = 1:length(constraint_labels),
    Constraint_Name = constraint_labels,
    Shadow_Price = solution$auxiliary$dual
  )


} else {
  cat(sprintf("Optimal solution NOT found. Solver status: %d\n", solution$status))
}
```

```
##   Bond_ID Type Units_Purchased Dollar_Value Percent_of_Bond_Portfolio
##        78    3        992517.3   1000000000                0.10740676
##        39    2        465404.7    560109947                0.06015959
##        77    3        488140.8    484784835                0.05206917
##        40    2        401740.8    470583052                0.05054380
##        32    2        256994.5    405169741                0.04351797
##        28    2        371901.6    403141298                0.04330010
##        41    2        354733.4    401302843                0.04310264
##        29    2        311403.9    400104135                0.04297389
##        30    2        361877.5    398481454                0.04279960
##        24    2        354645.8    393862479                0.04230349
##        34    2        362369.1    391938418                0.04209683
##        26    2        249184.9    391432169                0.04204246
##        36    2        343544.3    361068473                0.03878119
##        35    2        278600.2    342680985                0.03680625
##        50    2        314493.4    321327357                0.03451273
##        37    2        265862.2    320106030                0.03438155
##        38    2        302440.3    312756557                0.03359217
##        44    2        272046.8    287874468                0.03091966
##        45    2        228646.9    241583782                0.02594773
##        82    3        236694.4    237295396                0.02548713
```

Initial Outlay

```
cat(sprintf("\nTotal Economic Cost (Z = Initial Outlay + c_80): $%.2f\n", total_cost_Z))
```

```
##
## Total Economic Cost (Z = Initial Outlay + c_80): $9310401151.72
```

```
cat(sprintf("  Initial Outlay (Bond Purchases + c_0): $%.2f\n", initial_bond_cost + initial_cash_c0))
```

```
##   Initial Outlay (Bond Purchases + c_0): $9310401151.72
```

```
cat(sprintf("    - Initial Bond Purchases (sum(x_i*P_i)): $%.2f\n", initial_bond_cost))
```

```
##     - Initial Bond Purchases (sum(x_i*P_i)): $9310401151.72
```

```
cat(sprintf("    - Initial Cash Deposit (c_0): $%.2f\n", initial_cash_c0))
```

```
##     - Initial Cash Deposit (c_0): $0.00
```

```
cat(sprintf("  Residual Cash (c_80): $%.2f\n", final_cash_c80))
```

```
##   Residual Cash (c_80): $0.00
```

Purchased Bond

```
purchases_filtered
```

```
##    Bond_ID Type Units_Purchased Dollar_Value Percent_of_Bond_Portfolio
## 78      78    3    992517.31273 1.000000e+09              1.074068e-01
## 39      39    2    465404.73710 5.601099e+08              6.015959e-02
## 77      77    3    488140.80280 4.847848e+08              5.206917e-02
## 40      40    2    401740.75566 4.705831e+08              5.054380e-02
## 32      32    2    256994.45062 4.051697e+08              4.351797e-02
## 28      28    2    371901.56649 4.031413e+08              4.330010e-02
## 41      41    2    354733.43705 4.013028e+08              4.310264e-02
## 29      29    2    311403.85960 4.001041e+08              4.297389e-02
## 30      30    2    361877.54047 3.984815e+08              4.279960e-02
## 24      24    2    354645.75179 3.938625e+08              4.230349e-02
## 34      34    2    362369.09924 3.919384e+08              4.209683e-02
## 26      26    2    249184.94387 3.914322e+08              4.204246e-02
## 36      36    2    343544.27928 3.610685e+08              3.878119e-02
## 35      35    2    278600.16184 3.426810e+08              3.680625e-02
## 50      50    2    314493.41512 3.213274e+08              3.451273e-02
## 37      37    2    265862.17096 3.201060e+08              3.438155e-02
## 38      38    2    302440.31739 3.127566e+08              3.359217e-02
## 44      44    2    272046.78605 2.878745e+08              3.091966e-02
## 45      45    2    228646.93792 2.415838e+08              2.594773e-02
## 82      82   3    236694.40480 2.372954e+08             2.548713e-02
## 46      46   2    205120.65128 2.161172e+08             2.321244e-02
## 43      43   2    100978.88963 2.064847e+08             2.217785e-02
## 42      42   2    165442.43307 1.784280e+08             1.916437e-02
## 47      47   2    151811.68597 1.612787e+08             1.732242e-02
## 48      48   2    120754.48453 1.269939e+08             1.364000e-02
## 49      49   2    111152.45147 1.179372e+08             1.266725e-02
## 21      21   2     13450.83506 2.196333e+07             2.359010e-03
## 1        1   1     11343.78474 1.000000e+07             1.074068e-03
## 2        2   1     12330.08786 1.000000e+07             1.074068e-03
## 3        3   1     13296.30909 1.000000e+07             1.074068e-03
## 4        4   1     14216.75987 1.000000e+07             1.074068e-03
## 5        5   1     15183.75677 1.000000e+07             1.074068e-03
## 6        6   1     16193.73246 1.000000e+07             1.074068e-03
## 7        7   1     17274.32879 1.000000e+07             1.074068e-03
## 8        8   1     18350.58154 1.000000e+07             1.074068e-03
## 9        9   1     19651.73533 1.000000e+07             1.074068e-03
## 10      10   1     20874.68307 1.000000e+07             1.074068e-03
## 11      11   1     22402.24946 1.000000e+07             1.074068e-03
## 12      12   1     23901.92879 1.000000e+07             1.074068e-03
## 13      13   1     25498.27762 1.000000e+07             1.074068e-03
## 14      14   1     27338.43008 1.000000e+07             1.074068e-03
## 51      51   2      6296.75519 9.584291e+06             1.029418e-03
## 52      52   2      2771.25264 4.133767e+06             4.439945e-04
## 53      53   2      1119.18277 1.289455e+06             1.384962e-04
## 54      54   2       388.51727 4.219958e+05             4.532520e-05
## 55      55   2       113.85595 1.372579e+05             1.474243e-05
## 56      56   2        26.91388 2.806768e+04             3.014659e-06
```

Portfolio Credit Quality

```
cat(sprintf("Dollar-Weighted Avg. Credit Score: %.4f (Limit: %.1f)\n", avg_credit, credit_limit))
```

```
## Dollar-Weighted Avg. Credit Score: 0.0354 (Limit: 0.4)
```

Bond Allocation

```
  cat(sprintf("Allocation CIPS: %.2f%% (Limit: 30%%)\n", alloc_cips * 100))
```

```
## Allocation CIPS: 1.50% (Limit: 30%)
```

```
  cat(sprintf("Allocation TIPS: %.2f%% (Limit: 80%%)\n", alloc_tips * 100))
```

```
## Allocation TIPS: 80.00% (Limit: 80%)
```

```
  cat(sprintf("Allocation Nominal: %.2f%% (Limit: 80%%)\n", alloc_nom * 100))
```

```
## Allocation Nominal: 18.50% (Limit: 80%)
```

```r
  cat(sprintf("Allocation SMAs: %.2f%% (Limit: 10%%)\n", alloc_smas * 100))
```

```
## Allocation SMAs: 0.00% (Limit: 10%)
```

Portfolio Cash Flow Ratio

```r
cat(sprintf("Nominal CF Ratio: %.2f%% (Limit: 20%%)\n", nom_cf_ratio * 100))
```

```
## Nominal CF Ratio: 20.00% (Limit: 20%)
```

Shadow Prices of each Constraint

```r
duals_df
```

```
##    Constraint_ID      Constraint_Name Shadow_Price
## 1              1           Benefit_Q1    0.9766207
## 2              2           Benefit_Q2    0.9790408
## 3              3           Benefit_Q3    0.9814669
## 4              4           Benefit_Q4    0.9838990
## 5              5           Benefit_Q5    0.9863372
## 6              6           Benefit_Q6    0.9887814
## 7              7           Benefit_Q7    0.9912317
## 8              8           Benefit_Q8    0.9888662
## 9              9           Benefit_Q9    0.9755779
## 10            10          Benefit_Q10    0.9756218
## 11            11          Benefit_Q11    0.9762332
## 12            12          Benefit_Q12    0.9735816
## 13            13          Benefit_Q13    0.9586969
## 14            14          Benefit_Q14    0.9594518
## 15            15          Benefit_Q15    0.9611071
## 16            16          Benefit_Q16    0.9562362
## 17            17          Benefit_Q17    0.9477099
## 18            18          Benefit_Q18    0.9407567
## 19            19          Benefit_Q19    0.9422831
## 20            20          Benefit_Q20    0.9446181
## 21            21          Benefit_Q21    0.9274869
## 22            22          Benefit_Q22    0.9297853
## 23            23          Benefit_Q23    0.9201253
## 24            24          Benefit_Q24    0.9224055
## 25            25          Benefit_Q25    0.9030032
## 26            26          Benefit_Q26    0.8969290
## 27            27          Benefit_Q27    0.8932693
## 28            28          Benefit_Q28    0.8954829
## 29            29          Benefit_Q29    0.8732774
## 30            30          Benefit_Q30    0.8754414
## 31            31          Benefit_Q31    0.8636844
## 32            32          Benefit_Q32    0.8658246
## 33            33          Benefit_Q33    0.8440220
## 34              34          Benefit_Q34    0.8461135
## 35              35          Benefit_Q35    0.8346767
## 36              36          Benefit_Q36    0.8367451
## 37              37          Benefit_Q37    0.8157889
## 38              38          Benefit_Q38    0.8178104
## 39              39          Benefit_Q39    0.8074910
## 40              40          Benefit_Q40    0.8094920
## 41              41          Benefit_Q41    0.8114980
## 42              42          Benefit_Q42    0.8135089
## 43              43          Benefit_Q43    0.8155248
## 44              44          Benefit_Q44    0.8175458
## 45              45          Benefit_Q45    0.8195717
## 46              46          Benefit_Q46    0.8216026
## 47              47          Benefit_Q47    0.8236386
## 48              48          Benefit_Q48    0.8256796
## 49              49          Benefit_Q49    0.8277257
## 50              50          Benefit_Q50    0.8297769
## 51              51          Benefit_Q51    0.8318331
## 52              52          Benefit_Q52    0.8338945
## 53              53          Benefit_Q53    0.8359609
## 54              54          Benefit_Q54    0.8380325
## 55              55          Benefit_Q55    0.8401092
## 56              56          Benefit_Q56    0.8421910
## 57              57          Benefit_Q57    0.6813772
## 58              58          Benefit_Q58    0.6830657
## 59              59          Benefit_Q59    0.6847584
## 60              60          Benefit_Q60    0.6864553
## 61              61          Benefit_Q61    0.6530275
## 62              62          Benefit_Q62    0.6546458
## 63              63          Benefit_Q63    0.6562680
## 64              64          Benefit_Q64    0.6578943
## 65              65          Benefit_Q65    0.6274576
## 66              66          Benefit_Q66    0.6290124
## 67              67          Benefit_Q67    0.6305712
## 68              68          Benefit_Q68    0.6321338
## 69              69          Benefit_Q69    0.6037483
## 70              70          Benefit_Q70    0.6052444
## 71              71          Benefit_Q71    0.6067442
## 72              72          Benefit_Q72    0.6082478
## 73              73          Benefit_Q73    0.5705039
## 74              74          Benefit_Q74    0.5719177
## 75              75          Benefit_Q75    0.5733349
## 76              76          Benefit_Q76    0.5747557
## 77              77          Benefit_Q77    0.5517086
## 78              78          Benefit_Q78    0.5530758
## 79              79          Benefit_Q79    0.5544464
## 80              80          Benefit_Q80    0.5558203
## 81              81 Credit_Quality_Limit    0.0000000
## 82              82     Alloc_CIPS_Limit    0.0000000
## 83              83     Alloc_TIPS_Limit   -1.2500000
## 84              84  Alloc_Nominal_Limit    0.0000000
## 85              85     Alloc_SMAs_Limit    0.0000000
## 86              86     Nominal_CF_Limit   -1.2116542
```