# Selecting the Optimal Credit Card Portfolio

## Part 3: Progress Update

Remco A. Scheepmaker

26 June 2024

# Outline

# Introduction

# Recap

- **Goal**: Recommend an optimized credit card portfolio, based on user income/spend and preferences, and study its properties
- **Last Week**: Credit Card and Budget Data
- **This Week**: Results 1/2
  - Algorithm
  - Sensitivity Analysis
- In **Two Weeks**: Results 2/2
  - Monte Carlo Simulations
  - Shiny App

# Algorithm

# Two R Functions

1. `get_budget`: returns a budget based on income
2. `get_portfolio`: returns a portfolio (card names, net benefit, marginal benefit, return-on-spend, total spend, and which card to use for which category)
   - 0. Calculate "spend matrix": $y_{kc} = x_{kc} m_{kc} (\eta v_{t,k} + (1 - \eta) v_{b,k})$
   - 1. Calculate row sums:

$$\sum_{c=1}^{18} y_{kc} + \theta b_k - f_k$$

   - 2. Select highest value card, store its name, net benefit, marginal benefit, and spend categories
   - 3. Subtract its values from all cards in the matrix (set negative values to 0)
   - 4. Set the benefits of the selected card to 0! (avoids reselection)
   - Go to step 1 ($K$ times for $K$ cards)

# Tweaked Credit Card Dataset

- Find duplicates of selected card by ID number

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | Chase | Freedom Unlimited | 0 | 0 | TRUE | 0.01 | 0.01 | 1.5 | 0 | 5 | 0 | 5 | 0 |
| 9 | Chase | Sapphire Preferred | 95 | 50 | FALSE | 0.0125 | 0.026 | 2.1 | 0 | 5.1 | 0 | 5.1 | 0 |
| 10 | Chase | Sapphire Reserve | 550 | 360 | FALSE | 0.015 | 0.027 | 3 | 0 | 10 | 0 | 10 | 0 |
| 11 | Chase | Amazon Prime | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 5 | 0 | 5 | 0 |
| 12 | BoA | Premium Rewards | 95 | 120 | TRUE | 0.01 | 0.01 | 2 | 0 | 2 | 0 | 2 | 0 |
| 13 | BoA | Travel Rewards | 0 | 0 | FALSE | 0.01 | 0.01 | 1.5 | 0 | 3 | 0 | 3 | 0 |
| 14 | BoA | Unlimited Rewards | 0 | 0 | TRUE | 0.01 | 0.01 | 1.5 | 0 | 1.5 | 0 | 1.5 | 0 |
| 15 | BoA | Customized Cash Dining | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 16 | BoA | Customized Cash Online | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 17 | BoA | Customized Cash Gas | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 18 | BoA | Customized Cash Home | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 19 | BoA | Customized Cash Drug | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 20 | Citi / Wells Fargo | Double/Active Cash | 0 | 0 | TRUE | 0.01 | 0.01 | 2 | 0 | 2 | 0 | 2 | 0 |
| 21 | Citi | Strata Premier | 95 | 0 | FALSE | 0.01 | 0.015 | 3 | 0 | 10 | 0 | 10 | 0 |
| 22 | Citi | Custom Cash Gas | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 22 | Citi | Custom Cash Groceries | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 22 | Citi | Custom Cash Dining | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 22 | Citi | Custom Cash Streaming | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 22 | Citi | Custom Cash Drugs | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 22 | Citi | Custom Cash Home | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 22 | Citi | Custom Cash Entertainment | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 23 | CapOne | Venture X | 395 | 460 | FALSE | 0.01 | 0.017 | 2 | 0 | 5 | 0 | 10 | 0 |
| 24 | CapOne | Venture | 95 | 20 | FALSE | 0.01 | 0.017 | 2 | 0 | 2 | 0 | 5 | 0 |
| 25 | CapOne | Venture One | 0 | 0 | FALSE | 0.01 | 0.017 | 1.25 | 0 | 1.25 | 0 | 5 | 0 |
| 26 | CapOne | Savor | 95 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |
| 27 | CapOne | Savor One | 0 | 0 | TRUE | 0.01 | 0.01 | 1 | 0 | 1 | 0 | 1 | 0 |

# R Code for `get_budget`

```r
get_budget <- function (income = "avg", budget_data) {
  # Given an income (before taxes), this function returns a named
  # vector with the spend per category.
  # "avg" is accepted as a special input argument to return the average budget
  # from the Bureau of Labor Statistic's Consumer Expenditure Survey (2022),
  # corresponding to an average income of $94,003.

  if (income == "avg") {
    budget <- t(budget_data[2:19, 2])
  } else if (income < 15000) {
    budget <- t(budget_data[2:19, 5] * income)
  } else if (income >= 15000 & income < 30000) {
    budget <- t(budget_data[2:19, 7] * income)
  } else if (income >= 30000 & income < 40000) {
    budget <- t(budget_data[2:19, 9] * income)
  } else if (income >= 40000 & income < 50000) {
    budget <- t(budget_data[2:19, 11] * income)
  } else if (income >= 50000 & income < 70000) {
    budget <- t(budget_data[2:19, 13] * income)
  } else if (income >= 70000 & income < 100000) {
    budget <- t(budget_data[2:19, 15] * income)
  } else if (income >= 100000 & income < 150000) {
    budget <- t(budget_data[2:19, 17] * income)
  } else if (income >= 150000 & income < 200000) {
    budget <- t(budget_data[2:19, 19] * income)
  } else if (income >= 200000) {
    budget <- t(budget_data[2:19, 21] * income)
  }

  budget <- as.numeric(gsub('[$,]', '', budget))
  names(budget) <- budget_data[2:19,1]
  return(budget)
}
```

| Item | Inc_avg | Inc_avg_pct |
|---|---|---|
| gross_income | $94,003 | 100.00% |
| everything_else | $7,786 | 8.28% |
| groceries | $6,362 | 6.77% |
| dining | $4,222 | 4.49% |
| gas | $3,120 | 3.32% |
| utility | $3,117 | 3.32% |
| home_improvement | $2,606 | 2.77% |
| online_shopping | $1,881 | 2.00% |
| drug_store | $1,481 | 1.58% |
| travel_other | $1,460 | 1.55% |
| phone | $1,431 | 1.52% |
| streaming | $1,020 | 1.09% |
| department_store | $973 | 1.03% |
| entertainment | $833 | 0.89% |
| cable_internet | $698 | 0.74% |
| hotel_portal | $644 | 0.68% |
| airline_portal | $423 | 0.45% |
| car_portal | $394 | 0.42% |
| office_supplies | $128 | 0.14% |

# R Code output for `get_portfolio`

- Conservative user input

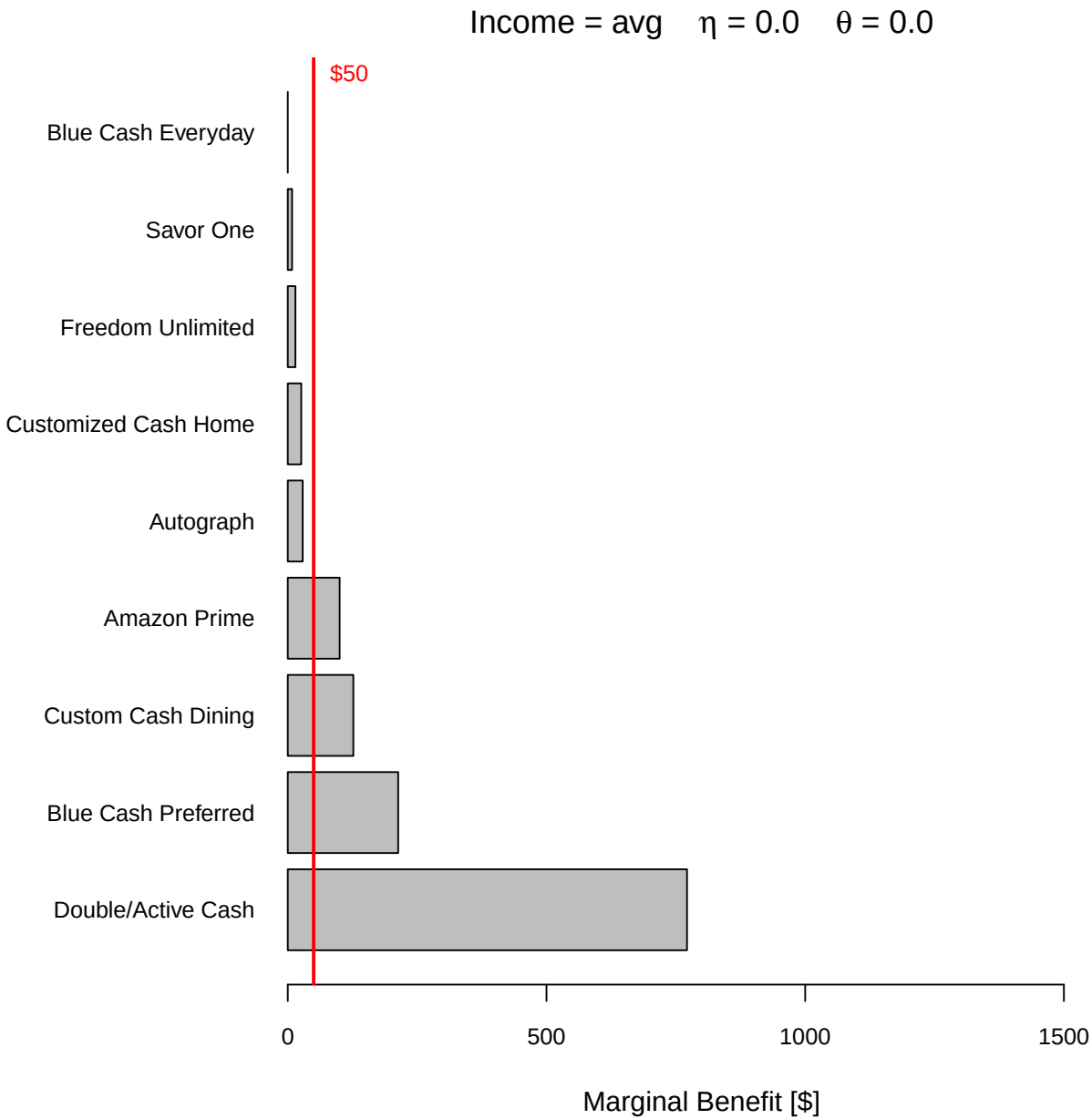
```
> income <- 'avg'
> K <- 4
> eta <- 0
> theta <- 0
> # # Create an optimal portfolio
> portfolio <- get_portfolio(income, K, eta, theta, cards_data, budget_data, verbose = TRUE)
[1] "The optimal portfolio with a total benefit of $1211.88 is:"
[1] "Double/Active Cash"  "Blue Cash Preferred" "Custom Cash Dining"  "Amazon Prime"
[1] "The marginal benefits are:"
[1] 771.58 213.38 126.66 100.26
[1] "The return on spend is: 3.14%"
[1] "Use the following card assignments:"
      everything_else            groceries               dining                  gas              utility
 "Double/Active Cash" "Blue Cash Preferred"  "Custom Cash Dining" "Blue Cash Preferred"  "Double/Active Cash"
    home_improvement       online_shopping           drug_store         travel_other                phone
 "Double/Active Cash"        "Amazon Prime"  "Double/Active Cash"  "Double/Active Cash"  "Double/Active Cash"
           streaming     department_store         entertainment       cable_internet         hotel_portal
"Blue Cash Preferred"  "Double/Active Cash"  "Double/Active Cash"  "Double/Active Cash"        "Amazon Prime"
      airline_portal            car_portal      office_supplies
       "Amazon Prime"        "Amazon Prime"  "Double/Active Cash"
> str(portfolio)
List of 6
 $ cards           : chr [1:4] "Double/Active Cash" "Blue Cash Preferred" "Custom Cash Dining" "Amazon Prime"
 $ net_benefit     : num [1:4] 772 985 1112 1212
 $ marginal_benefit: num [1:4] 772 213 127 100
 $ return_on_spend : num [1:4] 0.02 0.0255 0.0288 0.0314
 $ card_assignments: Named chr [1:18] "Double/Active Cash" "Blue Cash Preferred" "Custom Cash Dining" "Blue Ca
red" ...
  ..- attr(*, "names")= chr [1:18] "everything_else" "groceries" "dining" "gas" ...
 $ total_spend     : num 38579
```

# R Code output for `get_portfolio`

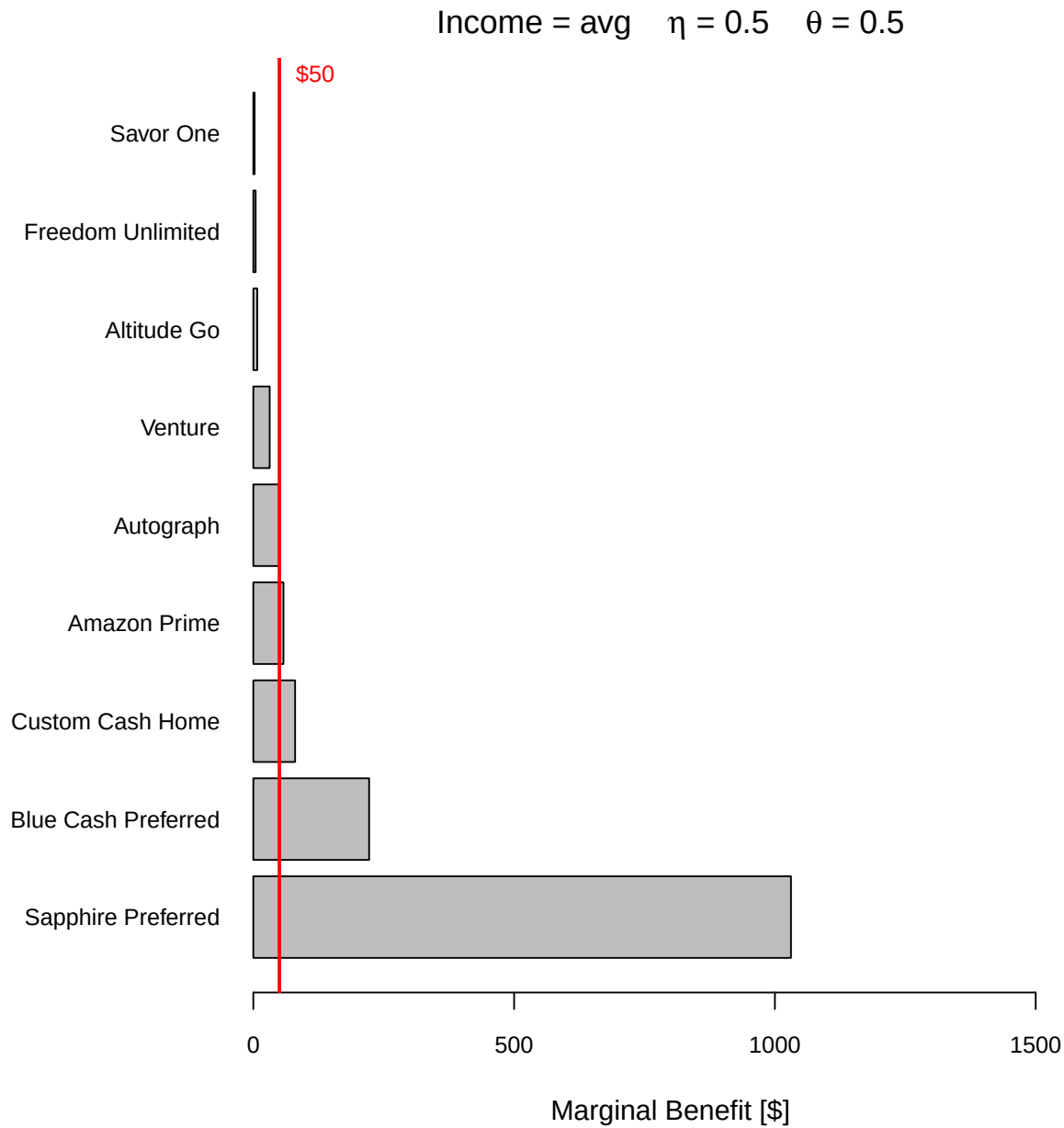
- Different user input, different output

```
> income <- 130000    #'avg'
> K <- 5
> eta <- 0.3
> theta <- 0.5
> # # Create an optimal portfolio
> portfolio <- get_portfolio(income, K, eta, theta, cards_data, budget_data, verbose = TRUE)
[1] "The optimal portfolio with a total benefit of $1700.04 is:"
[1] "Venture X"       "Gold"            "Custom Cash Gas" "Amazon Prime"    "Autograph"
[1] "The marginal benefits are:"
[1] 1199.97075  290.63661  112.97949   62.07919   34.37481
[1] "The return on spend is: 3.37%"
[1] "Use the following card assignments:"
  everything_else        groceries           dining              gas          utility  home_improvement
       "Venture X"            "Gold"           "Gold" "Custom Cash Gas"       "Venture X"       "Venture X"
  online_shopping       drug_store     travel_other            phone        streaming  department_store
    "Amazon Prime"       "Venture X"           "Gold"       "Autograph"       "Autograph"       "Venture X"
     entertainment    cable_internet     hotel_portal    airline_portal       car_portal   office_supplies
       "Venture X"       "Venture X"       "Venture X"       "Venture X"       "Venture X"       "Venture X"
> str(portfolio)
List of 6
 $ cards           : chr [1:5] "Venture X" "Gold" "Custom Cash Gas" "Amazon Prime" ...
 $ net_benefit     : num [1:5] 1200 1491 1604 1666 1700
 $ marginal_benefit: num [1:5] 1200 290.6 113 62.1 34.4
 $ return_on_spend : num [1:5] 0.0238 0.0296 0.0318 0.033 0.0337
 $ card_assignments: Named chr [1:18] "Venture X" "Gold" "Gold" "Custom Cash Gas" ...
  ..- attr(*, "names")= chr [1:18] "everything_else" "groceries" "dining" "gas" ...
 $ total_spend     : num 50409
```

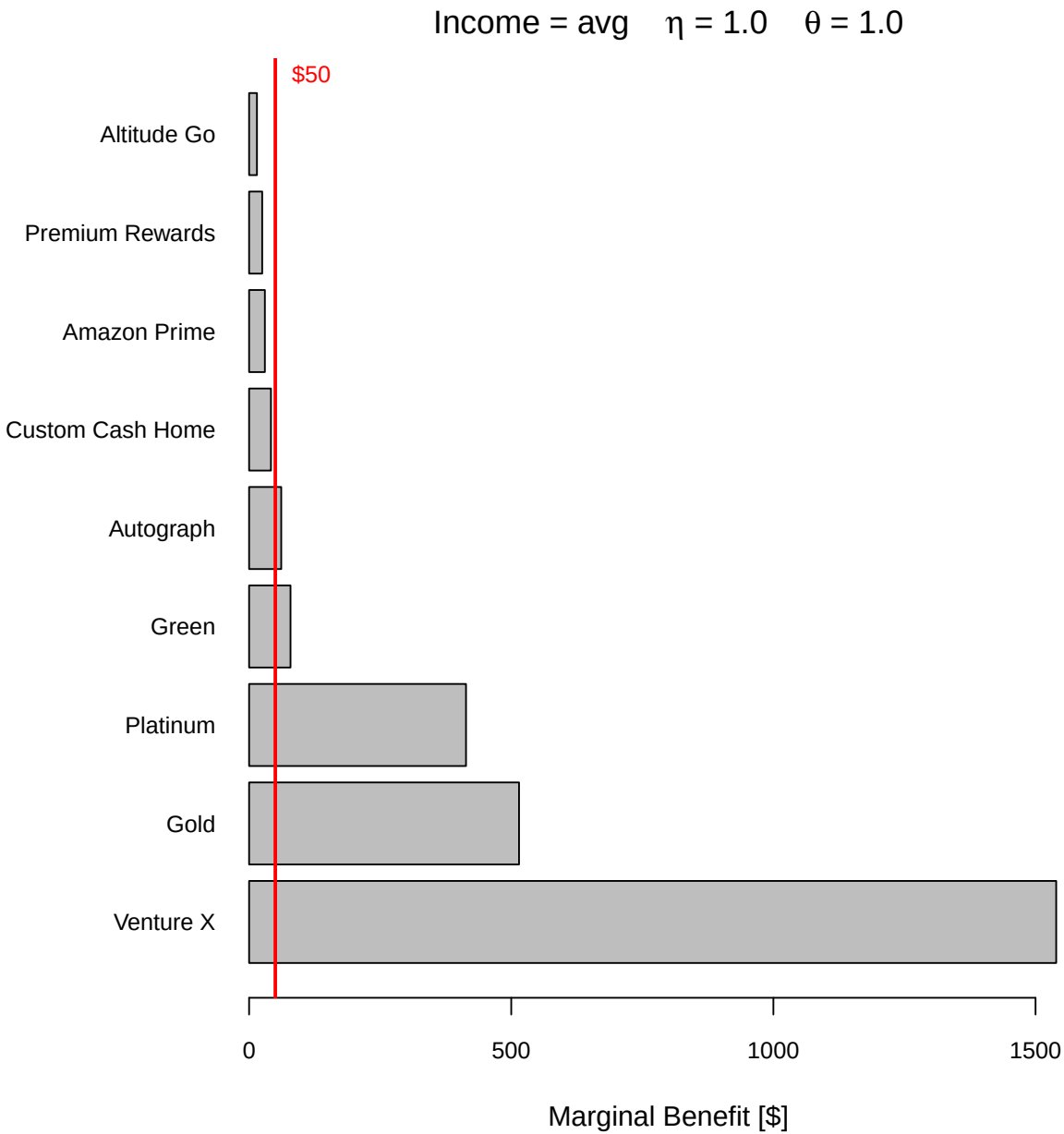
# Portfolios Visualized 1



# Portfolios Visualized 2

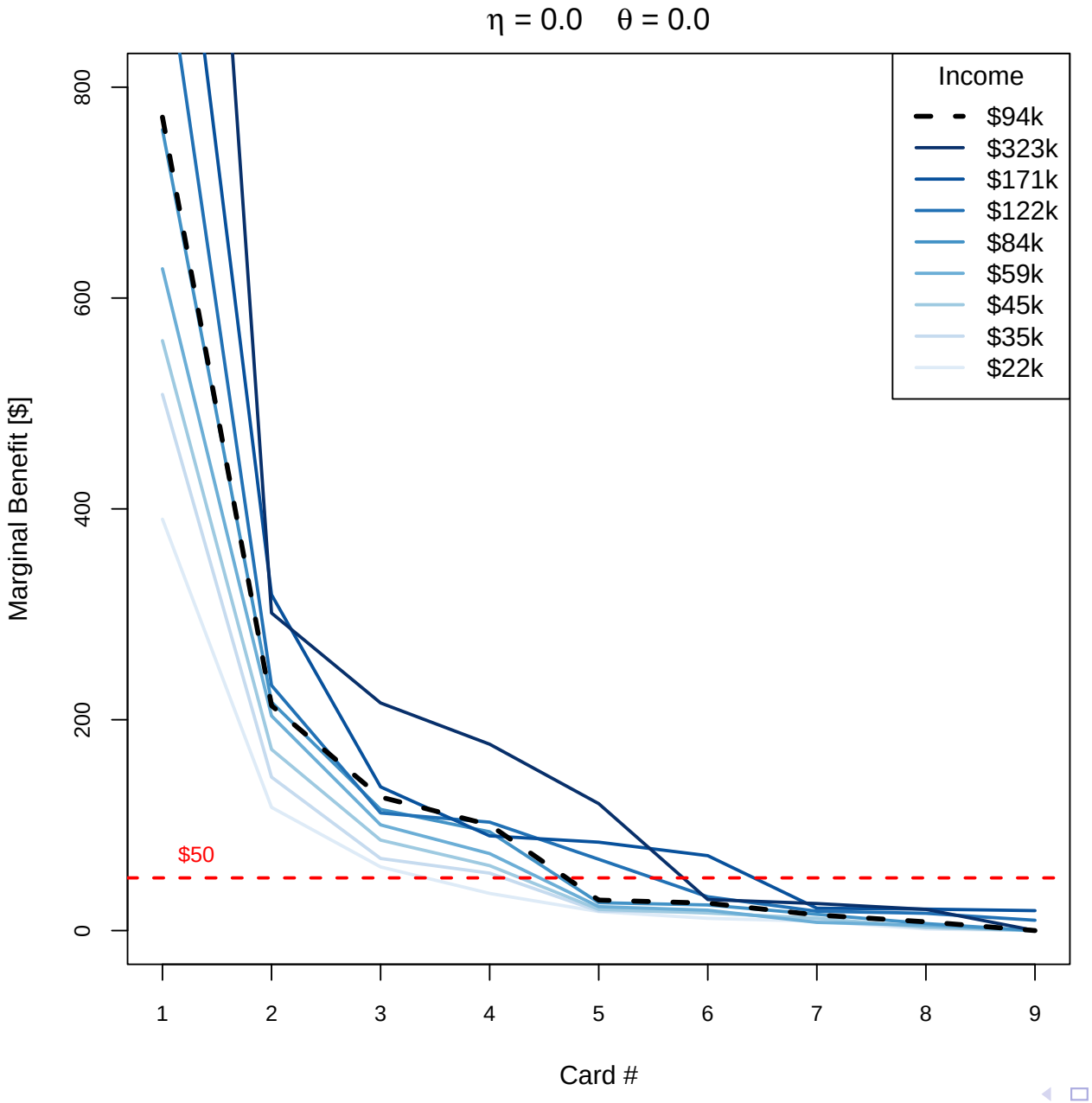


# Portfolios Visualized 3



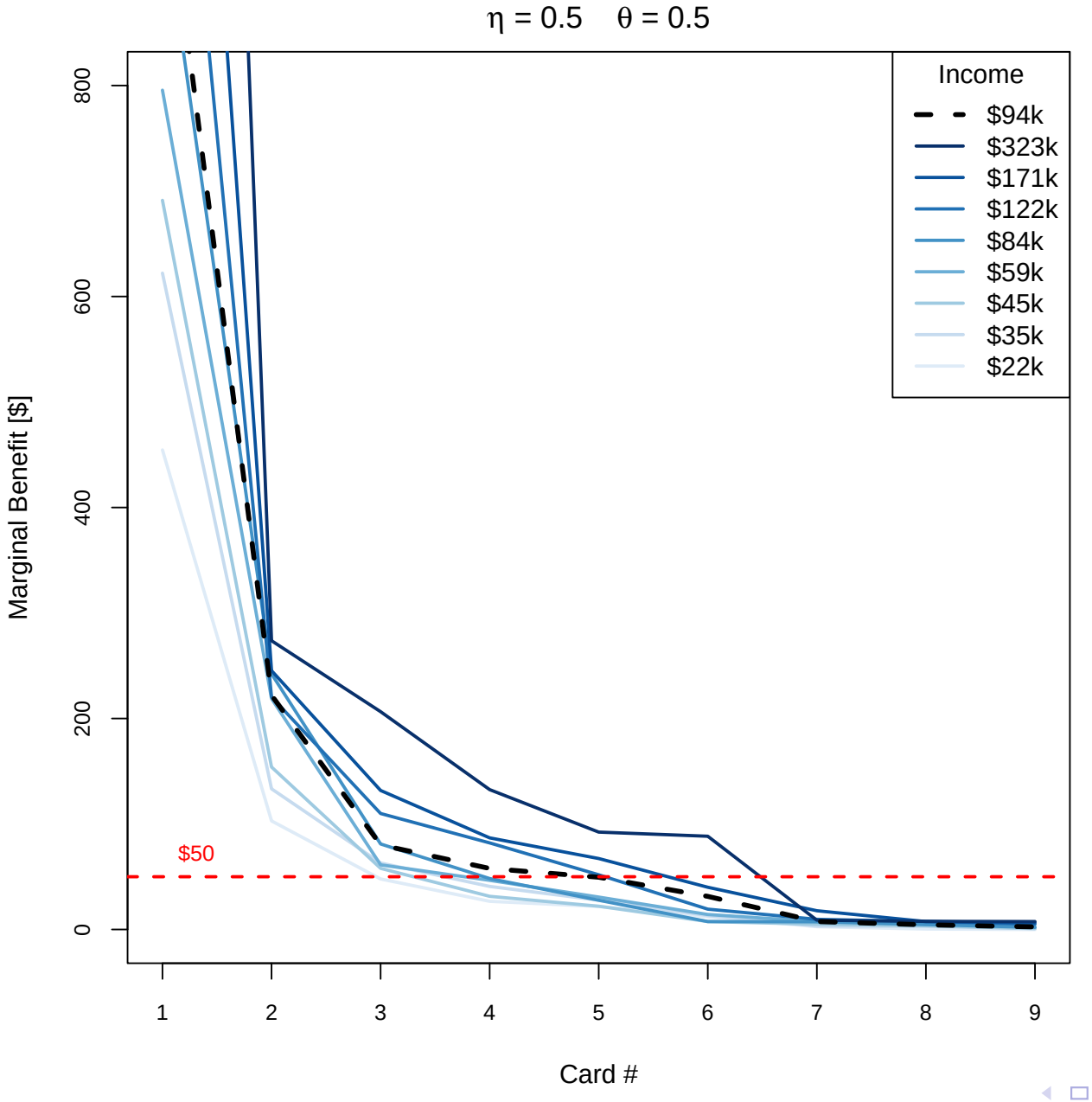
# Sensitivity Analysis

# Number of Cards



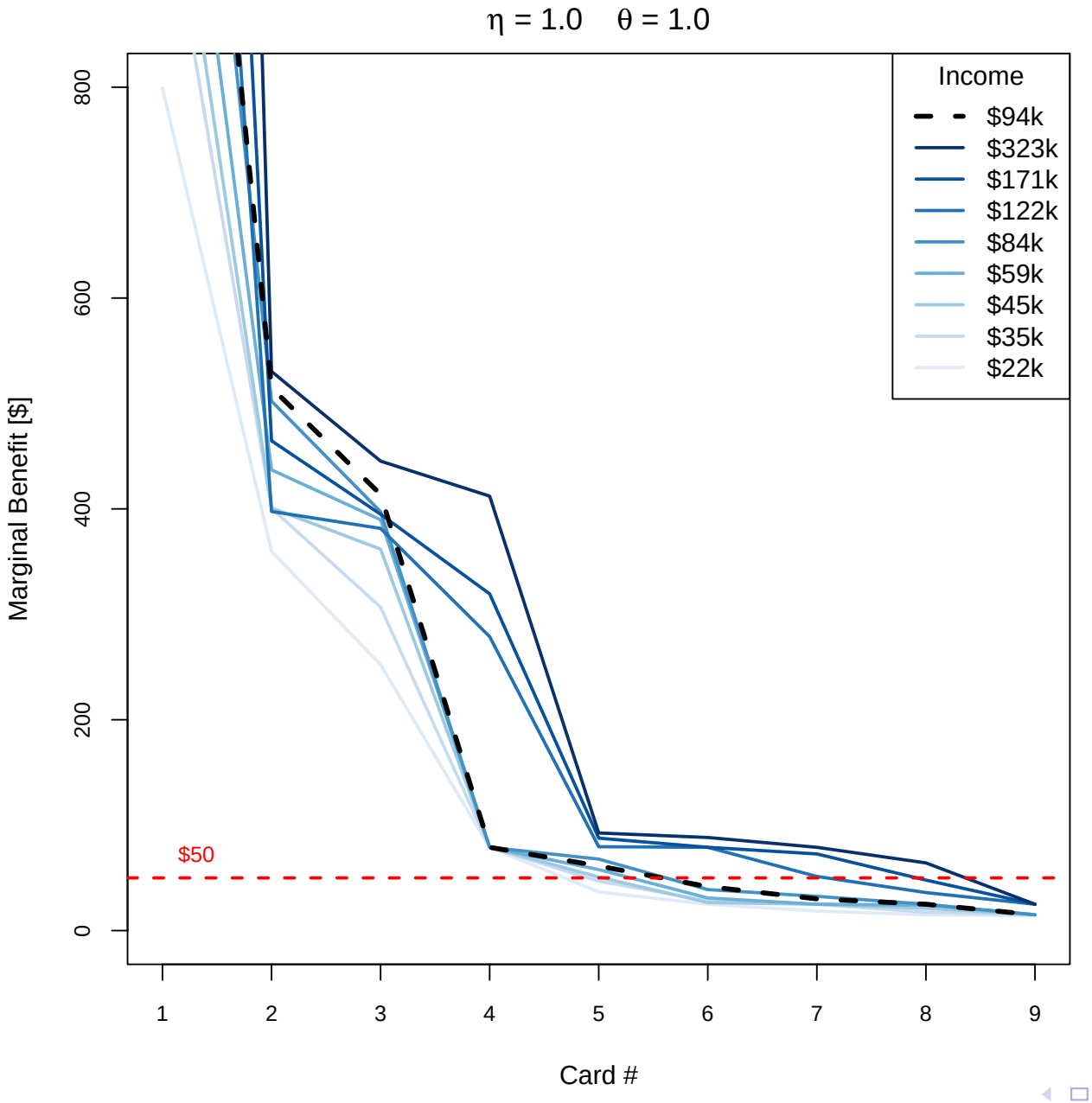

Four or five cards seems a reasonable choice for most people.

# Number of Cards




Four or five cards seems a reasonable choice for most people.

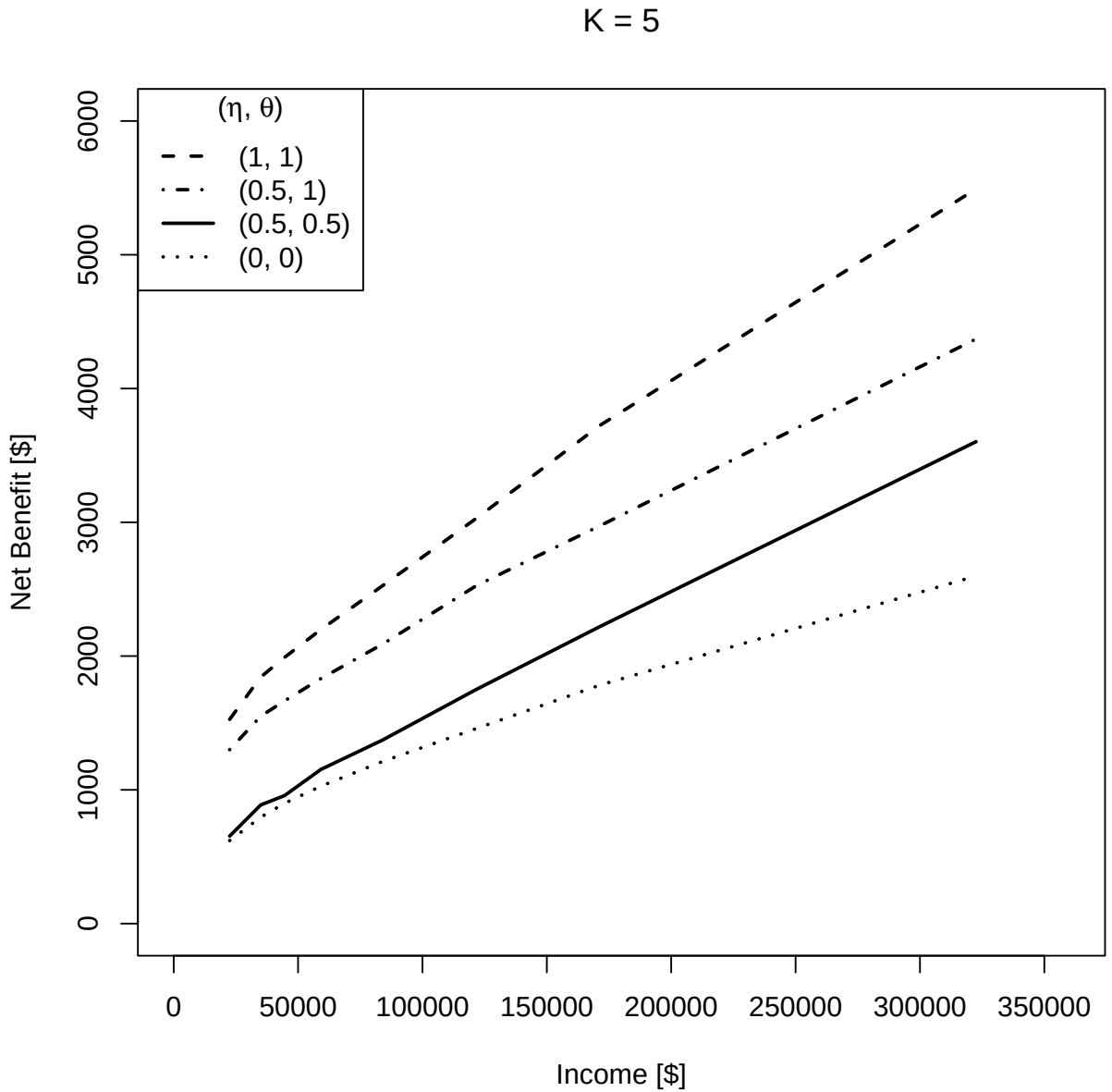
# Number of Cards




Wealthier people can justify annual fees more easily, unlocking better multipliers or benefits.

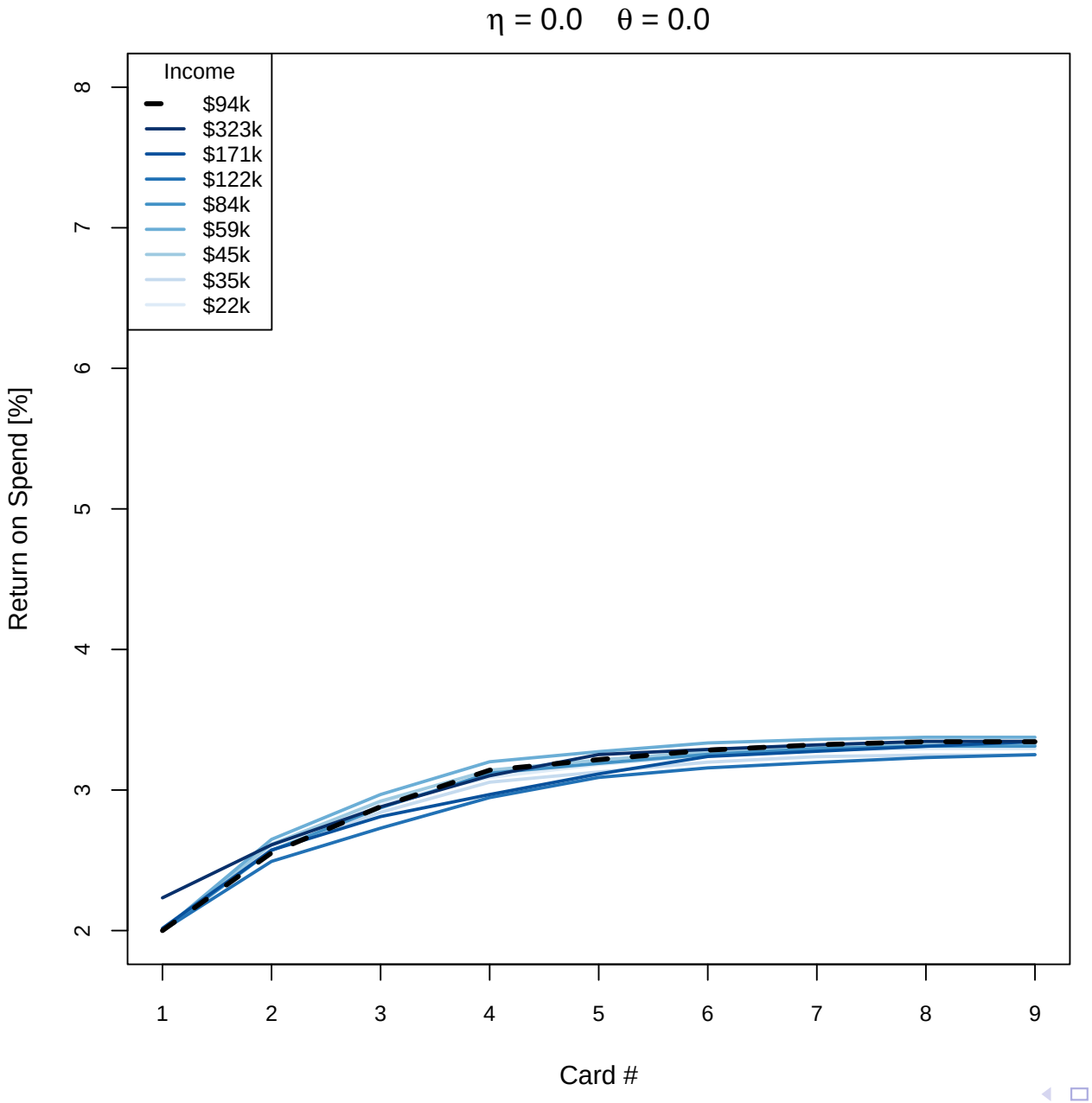
There is a potential issue with overlapping benefits.

# Net Benefit



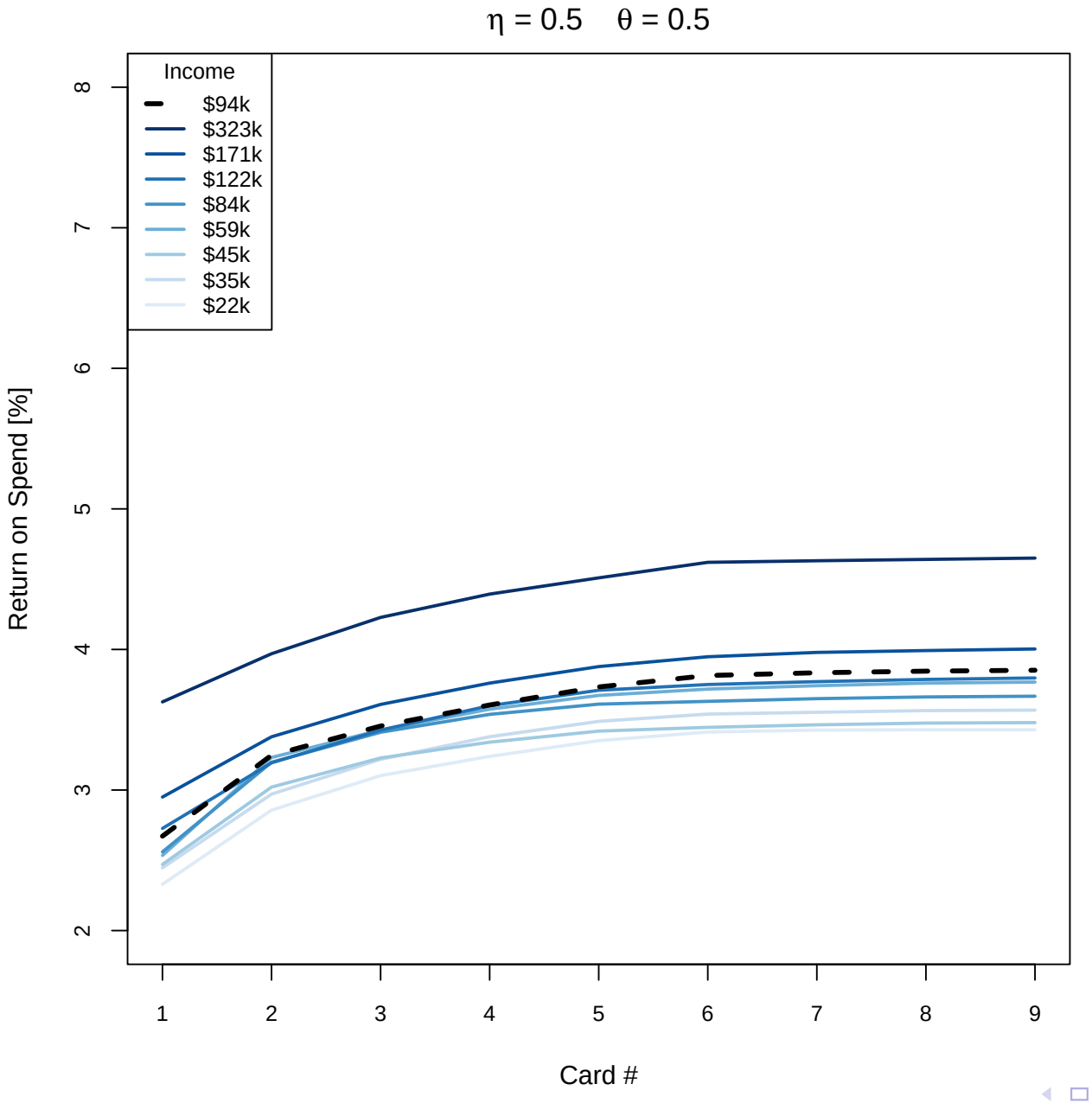

Shows the range of potential total benefit when using 5 cards.

# Return on Spend



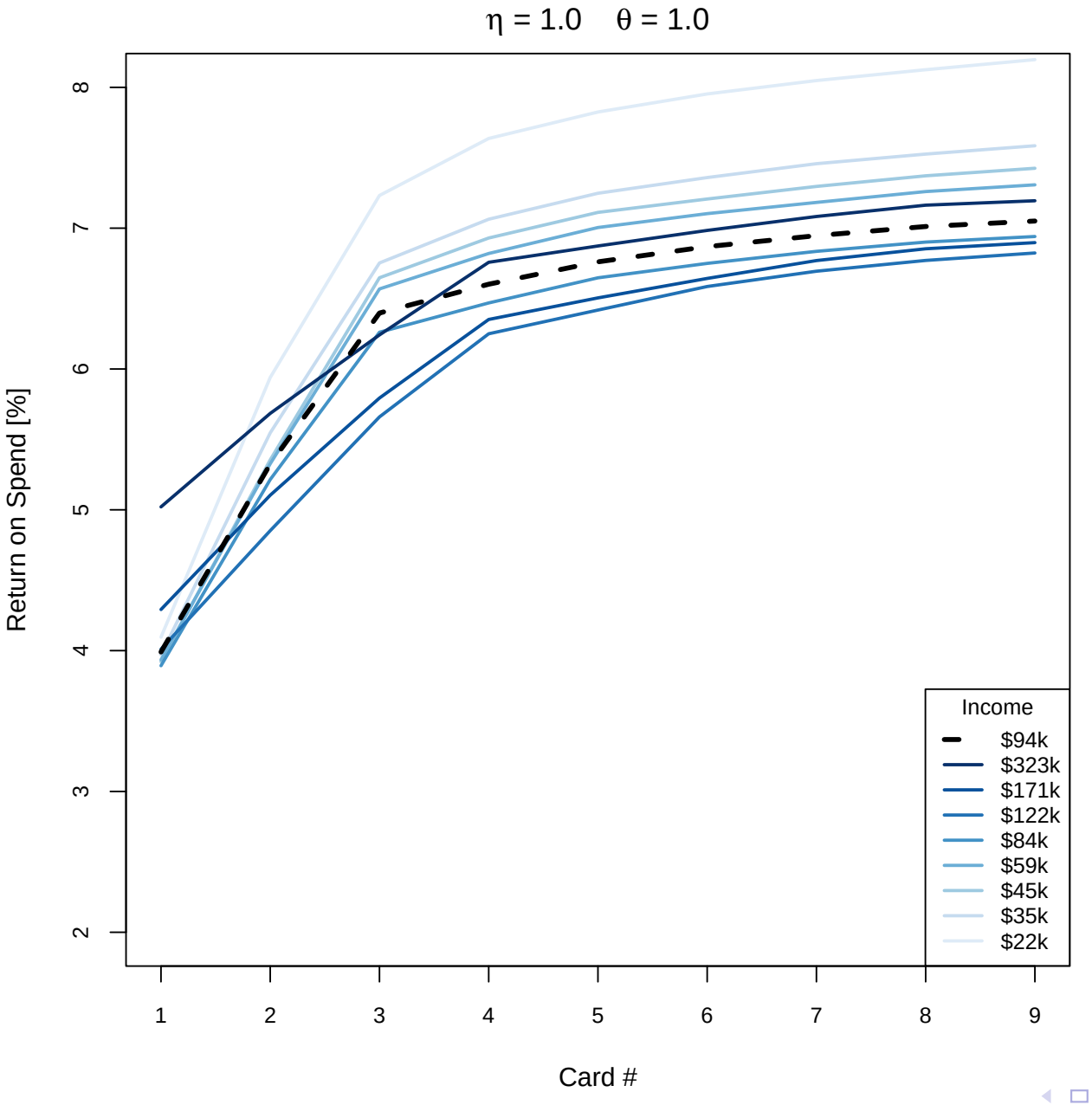

Everyone could increase their Return on Spend from 2% to 3.2–3.3% by using 5–6 cards.

# Return on Spend




Travel cards most benefit the wealthy.

# Return on Spend




Here we expose the limits of the model. . .

# We broke our model

- “Low incomes should just stack all the premium, high-fee travel cards and travel, using all the benefits multiple times!”

```
> income <- 22000
> K <- 4
> eta <- 1
> theta <- 1
> # # Create an optimal portfolio
> portfolio <- get_portfolio(income, K, eta, theta, cards_data, budget_data, verbose = TRUE)
[1] "The optimal portfolio with a total benefit of $1468.40 is:"
[1] "Platinum"  "Venture X" "Gold"      "Green"
[1] "The marginal benefits are:"
[1] 789.4506 353.4034 246.5450  79.0000
[1] "The return on spend is: 7.69%"
[1] "Use the following card assignments:"
 everything_else       groceries          dining             gas         utility home_improvement
      "Venture X"          "Gold"          "Gold"     "Venture X"     "Venture X"      "Venture X"
 online_shopping      drug_store    travel_other           phone       streaming department_store
      "Venture X"     "Venture X"       "Platinum"     "Venture X"     "Venture X"      "Venture X"
   entertainment   cable_internet    hotel_portal   airline_portal      car_portal
      "Venture X"     "Venture X"     "Venture X"       "Platinum"     "Venture X"
```

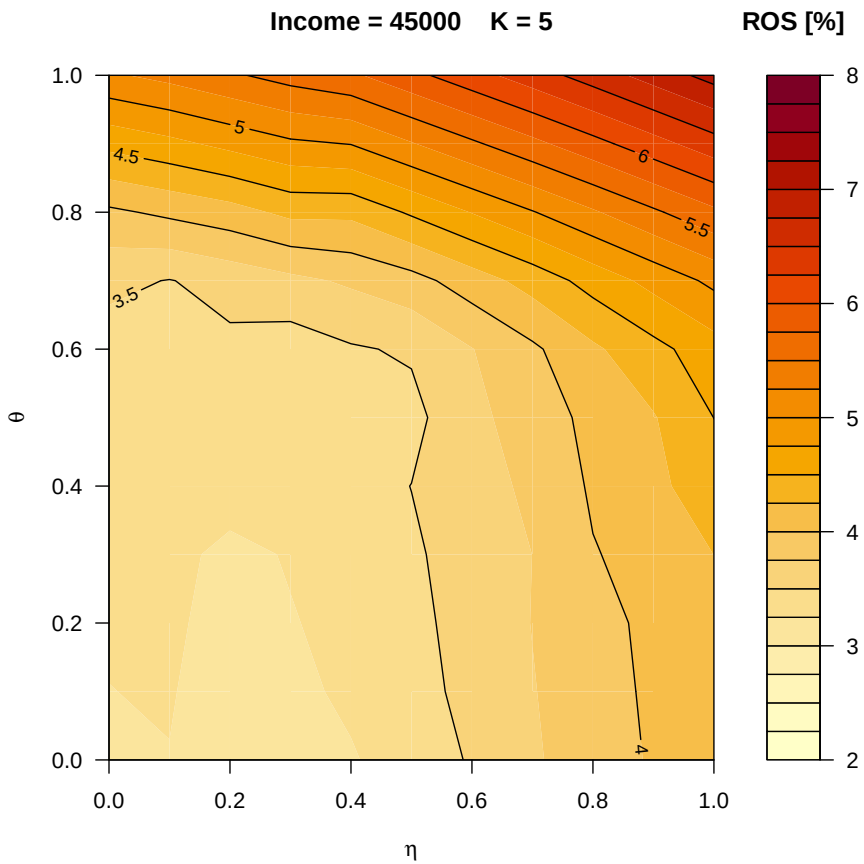
# We broke our model

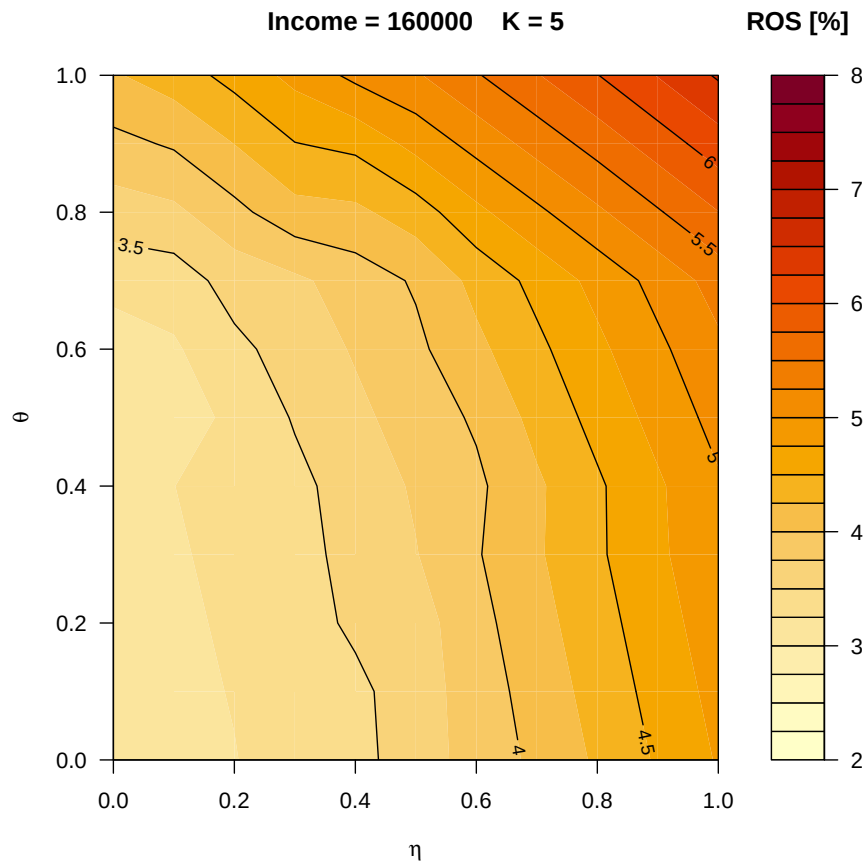
- "Low incomes should just stack all the premium, high-fee travel cards and travel, using all the benefits multiple times!"
- I will need to categorize the benefits and make them single-use…

```
> income <- 22000
> K <- 4
> eta <- 1
> theta <- 1
> # # Create an optimal portfolio
> portfolio <- get_portfolio(income, K, eta, theta, cards_data, budget_data, verbose = TRUE)
[1] "The optimal portfolio with a total benefit of $1468.40 is:"
[1] "Platinum"  "Venture X" "Gold"      "Green"
[1] "The marginal benefits are:"
[1] 789.4506 353.4034 246.5450  79.0000
[1] "The return on spend is: 7.69%"
[1] "Use the following card assignments:"
 everything_else       groceries          dining             gas         utility home_improvement
      "Venture X"            "Gold"          "Gold"      "Venture X"      "Venture X"      "Venture X"
 online_shopping      drug_store    travel_other           phone       streaming department_store
      "Venture X"      "Venture X"       "Platinum"      "Venture X"      "Venture X"      "Venture X"
   entertainment   cable_internet    hotel_portal   airline_portal      car_portal
      "Venture X"      "Venture X"      "Venture X"       "Platinum"      "Venture X"
```

# Return on Spend

- A different way of comparing two incomes
- All these figures need updating after fixing the benefits (easy using shell scripts!)



# Summary & Conclusions

# Summary & Conclusions

- Algorithm works, and produces realistic portfolios for a variety of inputs (when benefits are turned off)
- Spending on ~5 cards seems to be the sweet spot for most people (ROS of ~3.2%, assuming base valuations and no benefits)
- The Sensitivity Analysis exposed a flaw in the treatment of benefits, which can be fixed in the same way we deal with spend categories
- Next Steps: fix overlapping benefits, sample realistic incomes (Monte Carlo), build Shiny App

# Thank You!

# Extra Slides

# get_portfolio 1/3

```
# Get the budget corresponding to the income bin
budget <- get_budget(income, budget_data)
total_spend <- sum(budget)
# Get the category names (are sorted by highest average spend)
categories <- names(budget)
# Initialize the category spend value matrix
cat_value <- matrix(0, nrow = length(cards_data[,1]), ncol = length(categories))

# The portfolio that contains the selected cards in order
cards <- c()
# The net and marginal benefit with adding each card to the portfolio
net_benefit <- c()
marginal_benefit <- c()
# Return on Spend
# Named vector that tracks which card to use for every category
card_assignments <- c()

################################################
# Calculate Return Value of Spend per Card and Category
################################################
# K is the number of cards in the portfolio, while
# N is the number of cards in the dataset.

for (n in 1:length(cards_data[,1])) {
  for (c in 1:length(categories)) {
    cat <- categories[c]
    cap <- paste0(cat, "_cap")

    point_value <- eta * cards_data[n, "travel_value"] + (1 - eta) * cards_data[n, "base_value"]

    if (cards_data[n, cap] == 0) {
      # No spending cap, multiply spend * multiplier * value as usual:
      cat_value[n, c] <- budget[[cat]] * cards_data[n, cat] * point_value
    } else if (budget[[cat]] <= cards_data[n, cap]) {
      # There is a cap, but we stay below it:
      cat_value[n, c] <- budget[[cat]] * cards_data[n, cat] * point_value
    } else {
      # We spend more than the cap, so fall back to 1x multiplier beyond the cap
      cat_value[n, c] <- (cards_data[n, cap] * cards_data[n, cat] +
                            (budget[[cat]] - cards_data[n, cap]) ) * point_value
    }
  }
}
```

# get_portfolio 2/3

```
for (k in 1:K) {
  # Net benefit per card
  net_benefit_per_card <- rowSums(cat_value)  +
    theta * cards_data[, "benefits"] -
    cards_data[, "fee"]

  # Pick the card with max net_benefit_per_card:
  max_ind <- which.max(net_benefit_per_card)
  # Look up that card's name
  cardname <- cards_data[ max_ind , "name"]
  # Add the selected card to the portfolio
  cards <- append(cards, cardname )
  # Store the cumulative net benefit and marginal benefit
  # Note: net_benefit / sum(budget) would show the return on spend
  if (k == 1) {
    net_benefit <- append(net_benefit, net_benefit_per_card[ max_ind ])
  } else {
    net_benefit <- append(net_benefit, tail(net_benefit, n=1) + net_benefit_per_card[ max_ind ])
  }
  marginal_benefit <- append(marginal_benefit, net_benefit_per_card[ max_ind ])

  # Assign spending categories with additional value to the card:
  card_assignments[categories[cat_value[ max_ind ,] > 0] ] <- cardname

  # Subtract the selected card's values from the value matrix
  cat_value <- sweep(cat_value, 2, cat_value[max_ind, ])
  # And set negative categories to zero,
  cat_value[cat_value < 0] <- 0
  # Now we're left with positive categories that have remaining value that
  # we can retrieve with additional cards in the next iteration.

  # If the card can only be held once (Amex Platinum, Citi Custom Cash, etc.)
  # We need to make sure it will not be selected again for just the benefits
  # or additional rewards in other custom categories. This is accomplished by
  # setting all the values for the card, including benefits, to zero after
  # selection. We use ID to find duplicate cards that can only be held once.
  # If cards can be held multiple times (e.g. BoA Customized Cash), they will
  # have different ID numbers, if not, the duplicates will have the same ID.
  same_ind <- which(cards_data[,"id"] == cards_data[max_ind ,"id"])
  cards_data[same_ind, "benefits"] <- 0
  cat_value[same_ind, ] <- 0
}
```

# get_portfolio 3/3

```
  if (verbose == TRUE) {
    print(sprintf("The optimal portfolio with a total benefit of $%.2f is:", tail(net_benefit, n=1)))
    print(cards)
    print(sprintf("The marginal benefits are:"))
    print(marginal_benefit)
    print(sprintf("The return on spend is: %.2f%%", 100 * tail(net_benefit, n=1) / total_spend))
    print(sprintf("Use the following card assignments:"))
    print(card_assignments)
  }

  # Return a named list
  return(list("cards" = cards, "net_benefit" = net_benefit,
              "marginal_benefit" = marginal_benefit,
              "return_on_spend" = net_benefit / total_spend,
              "card_assignments" = card_assignments,
              "total_spend" = total_spend))
}
```