Term Project

Shujuan Chen,

Lei Chen,

Ziwei Yang,

Yibo Yan,

ECS 172: Recommender Systems

Norm Matloff

March 2022

Contents

1 Introduction **3**

2 Experimental Methods **3**

2.1 Notes on the algorithm 4

2.2 aEast Generation 5

2.2.1 InstEval 5

2.2.2 MovieLens 5

3 Experimental Results **6**

3.1 MovieLens 11

3.1.1 Linear Model 11

3.1.2 Matrix Factorization 21

3.2 InstEval 30

3.2.1 Linear Model 30

3.2.2 Matrix Factorization 40

4 Conclusion **51**

A Code **52**

B Team Member Contribution **65**

1 Introduction

In real-world collaborative filtering problems, we almost always encounter the three parameters: Number of users, Number of items, and the density of non-NA data. In this report, we try to investigate the effect of these three parameters on CF prediction accuracy.

2 Experimental Methods

The structure of our experiment is as follows:

Algorithm 1 Controlled Trail for d, m, n

```
for dataset = InstEval, MovieLens do
    form aEst from dataset using linear model
    for d = 20, 40, 60, 100 do
        M = total #of items

        if dataset == InstEval
        for m = 200, 600, 800, 1000, M do
            N = total #of Users
            for n = 200, 600, 800, 1000, N do
                create rating_matrix_subset of (m, n, d)
                for model = Linear, MatrixFactorization do
                    Split rating_matrix_subset into train(70%) and test set(30%)
                    train model on train set & record the run-time
                    test the model on test set & record MAPE
                end for
            end for
        end for

        if dataset == MovieLens
        for m = 300, 600, 900, 1200, M do
            N = total #of Users
            for n = 150, 300, 450, 600, 750, N do
                create rating_matrix_subset of (m, n, d)
                for model = Linear, MatrixFactorization do
                    Split rating_matrix_subset into train(70%) and test set(30%)
                    train model on train set & record the run-time
                    test the model on test set & record MAPE
                end for
            end for
        end for
    end for

end for
```

2.1 Notes on the algorithm

MAPE.: mean absolute percentage error

n: Number of users.

m: Number of items.

d: density of non-NA data.

-formula: # of non-NA values / n * m.

2.2 aEast Generation

Here we are experimenting on two of our old friends, InstEval and MovieLens. For both dataset, we generated two new columns: 'userMean' and 'itemMean'. We then train a linear model on 'user_mean', 'item_mean' to predict 'ratings'. aEast was generated by filling the NA values in the original dataset matrix using the trained linear model.

Here are some brief description of the datasets used.

2.2.1 InstEval

InstEval is University Lecture/Instructor Evaluations by Students at ETH Zurich. Detailed description of the dataset can be found here.

In running the code, the whole dataset did not finish after running 12 hours. Therefore, we decided to scaled down and test on half of the dataset(first 30000 lines). We believe 30k data is good enough for our purpose.

user: student, coded as 's' in the dataset
item: instructor, coded as 'd' in the dataset
userMean: mean rating a student gave
itemMean: mean rating a instructor receives

2.2.2 MovieLens

The data was collected through the MovieLens web site, where 943 users rated on 1682 movies. There is a total of 100,000 ratings on a scale of 1-5. Detailed description of the dataset could be found here.

user: user or rater, coded as 'user' in the dataset

item: movie, coded as 'item' in the dataset

userMean: mean rating a user gave

itemMean: mean rating a movie receives

3 Experimental Results

List of plots

1 d-20 (mape) . . . 12
2 d-20 (run time) . . . 12
3 d-40 (mape) . . . 12
4 d-40 (run time) . . . 12
5 d-60 (mape) . . . 13
6 d-60 (run time) . . . 13
7 d-80 (mape) . . . 13
8 d-80 (run time) . . . 13
9 d-100 (mape) . . . 14
10 d-100 (run time) . . . 14
11 m-300 (mape) . . . 15
12 m-300 (run time) . . . 15
13 m-600 (mape) . . . 15
14 m-600 (run time) . . . 15
15 m-900 (mape) . . . 16
16 m-900 (run time) . . . 16
17 m-1200 (mape) . . . 16

18 m-1200 (run time) 16
19 m-1682 (mape) 17
20 m-1682 (run time) 17
21 n-150 (mape) 18
22 n-150 (run time) 18
23 n-300 (mape) 18
24 n-300 (run time) 18
25 n-450 (mape) 19
26 n-450 (run time) 19
27 n-600 (mape) 19
28 n-600 (run time) 19
29 n-750 (mape) 20
30 n-750 (run time) 20
31 n-943 (mape) 20
32 n-943 (run time) 20
33 d-20 (mape) 21
34 d-20 (run time) 21
35 d-40 (mape) 22
36 d-40 (run time) 22
37 d-60 (mape) 22
38 d-60 (run time) 22
39 d-80 (mape) 23
40 d-80 (run time) 23
41 d-100 (mape) 23
42 d-100 (run time) 23
43 m-300 (mape) 24
44 m-300 (run time) 24

45 m-600 (mape) . . . 25
46 m-600 (run time) . . . 25
47 m-900 (mape) . . . 25
48 m-900 (run time) . . . 25
49 m-1200 (mape) . . . 26
50 m-1200 (run time) . . . 26
51 m-1682 (mape) . . . 26
52 m-1682 (run time) . . . 26
53 n-150 (mape) . . . 27
54 n-150 (run time) . . . 27
55 n-300 (mape) . . . 28
56 n-300 (run time) . . . 28
57 n-450 (mape) . . . 28
58 n-450 (run time) . . . 28
59 n-600 (mape) . . . 29
60 n-600 (run time) . . . 29
61 n-750 (mape) . . . 29
62 n-750 (run time) . . . 29
63 n-943 (mape) . . . 30
64 n-943 (run time) . . . 30
65 d-20 (mape) . . . 31
66 d-20 (run time) . . . 31
67 d-40 (mape) . . . 31
68 d-40 (run time) . . . 31
69 d-60 (mape) . . . 32
70 d-60 (run time) . . . 32
71 d-80 (mape) . . . 32

72 d-80 (run time) . . . 32
73 d-100 (mape) . . . 33
74 d-100 (run time) . . . 33
75 m-200 (mape) . . . 34
76 m-200 (run time) . . . 34
77 m-400 (mape) . . . 34
78 m-400 (run time) . . . 34
79 m-600 (mape) . . . 35
80 m-600 (run time) . . . 35
81 m-800 (mape) . . . 35
82 m-800 (run time) . . . 35
83 m-1000 (mape) . . . 36
84 m-1000 (run time) . . . 36
85 m-1128 (mape) . . . 36
86 m-1128 (run time) . . . 36
87 n-200 (mape) . . . 37
88 n-200 (run time) . . . 37
89 n-400 (mape) . . . 38
90 n-400 (run time) . . . 38
91 n-600 (mape) . . . 38
92 n-600 (run time) . . . 38
93 n-800 (mape) . . . 39
94 n-800 (run time) . . . 39
95 n-1000 (mape) . . . 39
96 n-1000 (run time) . . . 39
97 n-1216 (mape) . . . 40
98 n-1216 (run time) . . . 40

99 d-20 (mape) . . . 41
100 d-20 (run time) . . . 41
101 d-40 (mape) . . . 41
102 d-40 (run time) . . . 41
103 d-60 (mape) . . . 42
104 d-60 (run time) . . . 42
105 d-80 (mape) . . . 42
106 d-80 (run time) . . . 42
107 d-100 (mape) . . . 43
108 d-100 (run time) . . . 43
109 m-200 (mape) . . . 44
110 m-200 (run time) . . . 44
111 m-400 (mape) . . . 44
112 m-400 (run time) . . . 44
113 m-600 (mape) . . . 45
114 m-600 (run time) . . . 45
115 m-800 (mape) . . . 45
116 m-800 (run time) . . . 45
117 m-1000 (mape) . . . 46
118 m-1000 (run time) . . . 46
119 m-1128 (mape) . . . 46
120 m-1128 (run time) . . . 46
121 n-200 (mape) . . . 47
122 n-200 (run time) . . . 47
123 n-400 (mape) . . . 48
124 n-400 (run time) . . . 48
125 n-600 (mape) . . . 48

126 n-600 (run time) 48
127 n-800 (mape) 49
128 n-800 (run time) 49
129 n-1000 (mape) 49
130 n-1000 (run time) 49
131 n-1216 (mape) 50
132 n-1216 (run time) 50

3.1 MovieLens

3.1.1 Linear Model

Fixed D

- Compare across curves:

  For all values of d, the curves of MAPE move upwards as M(number of items) increase. That is, MAPE decreases steadily and proportionally with M. And run time increases steadily with M increases.

- Compare along curves: For all values of d and M, increasing N will decrease MAPE.Increasing N will increase the run-time.

- Remark:

  For low values of d(20 and 40) and M(300 and 600), there is no significance improvement of MAPE as N increases. One possible explanation is that when the available item rating is too low, adding more users to the regression does not provide more information.

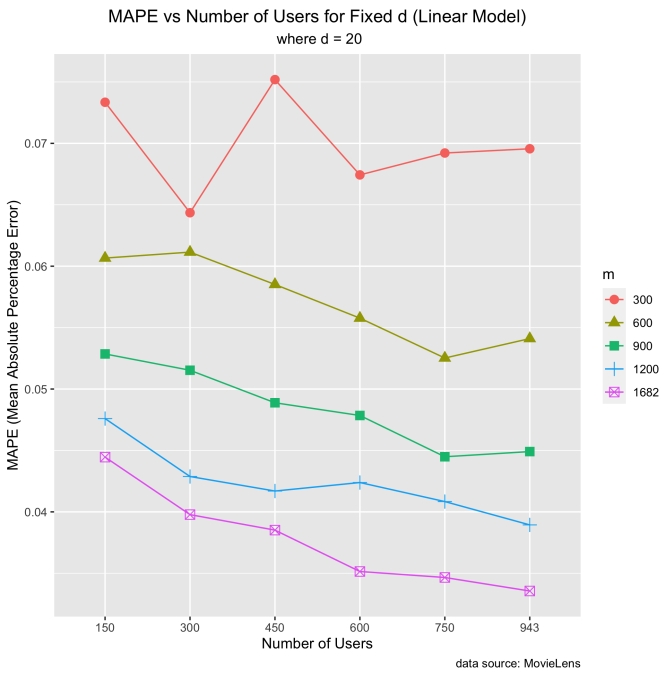

Figure 1: d-20 (mape)

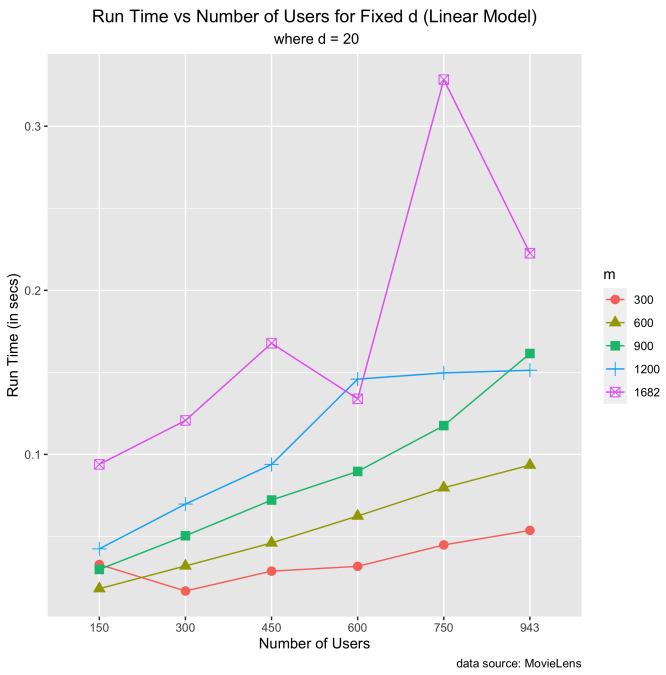

Figure 2: d-20 (run time)

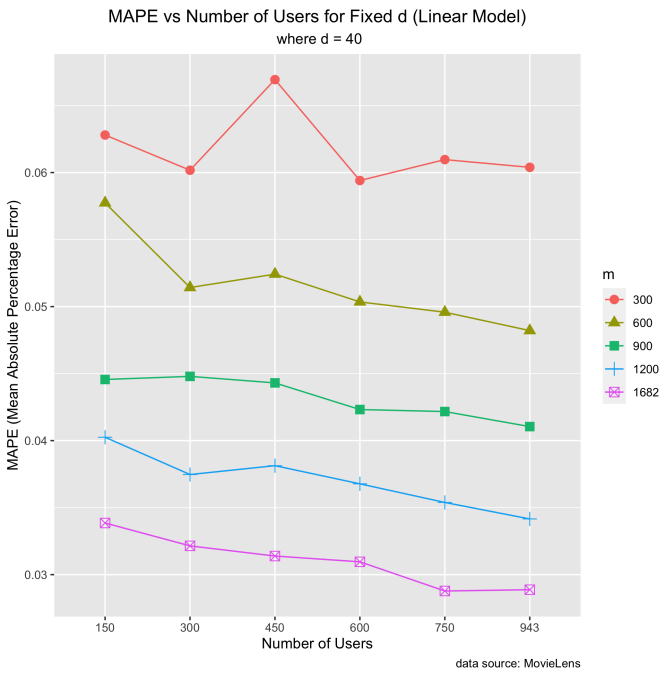

Figure 3: d-40 (mape)

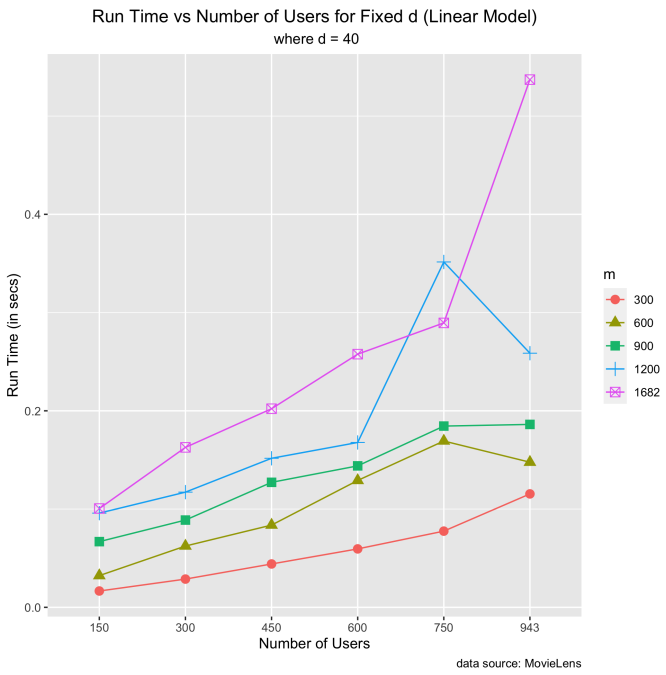

Figure 4: d-40 (run time)

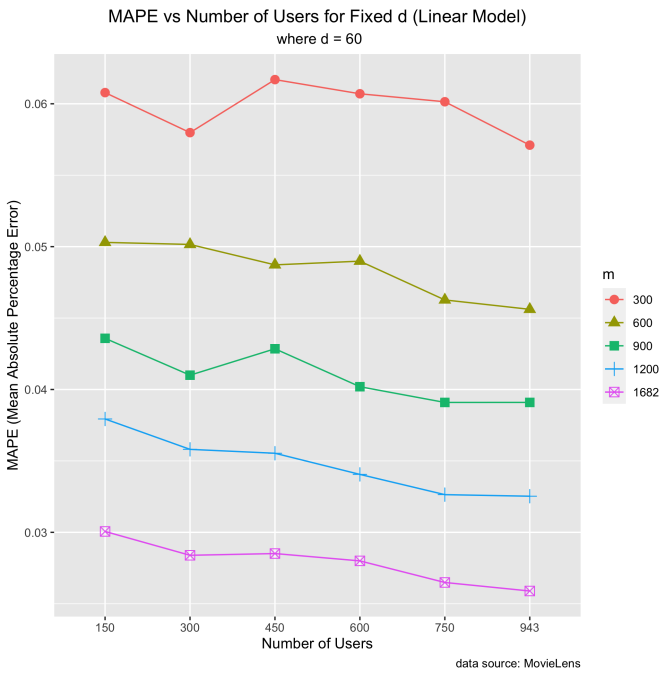

Figure 5: d-60 (mape)

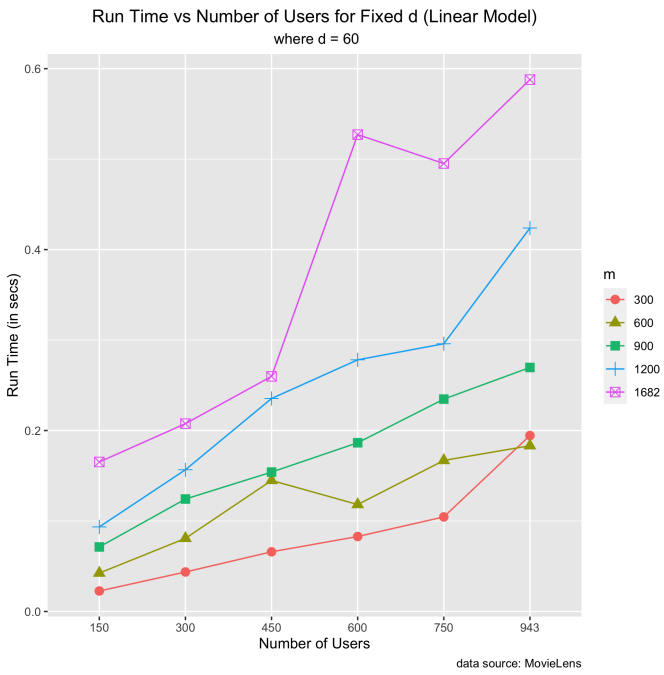

Figure 6: d-60 (run time)

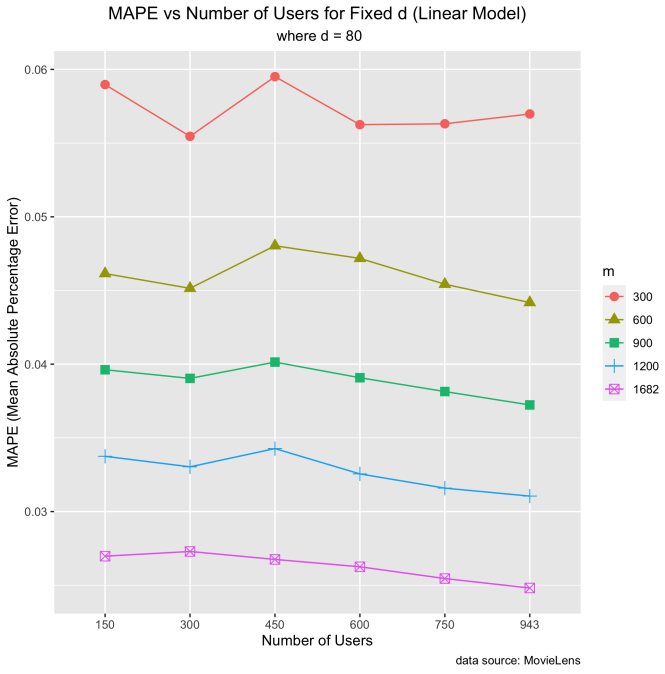

Figure 7: d-80 (mape)

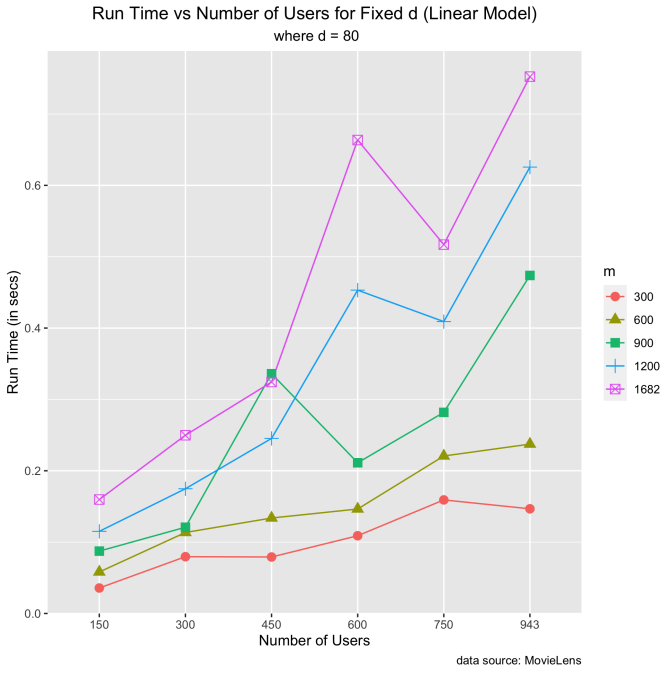

Figure 8: d-80 (run time)

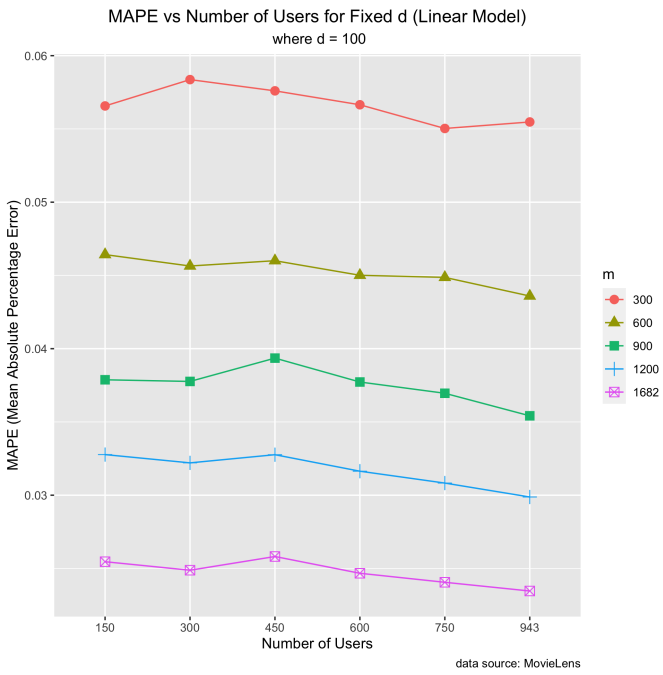

Figure 9: d-100 (mape)

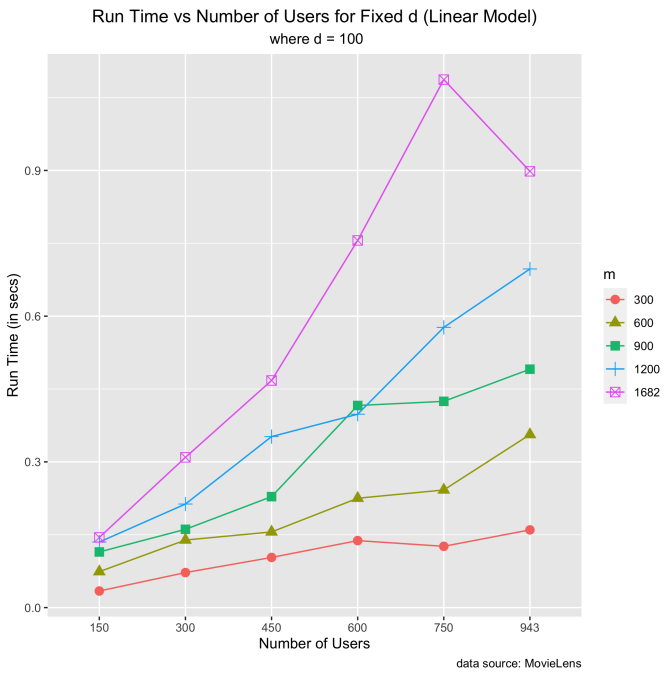

Figure 10: d-100 (run time)

Fixed M

- Compare across curves: As N increases, curves of MAPE move downward, curves of run time move upward.
- Compare along curves: As d increases, MAPE decreases and run time increases.

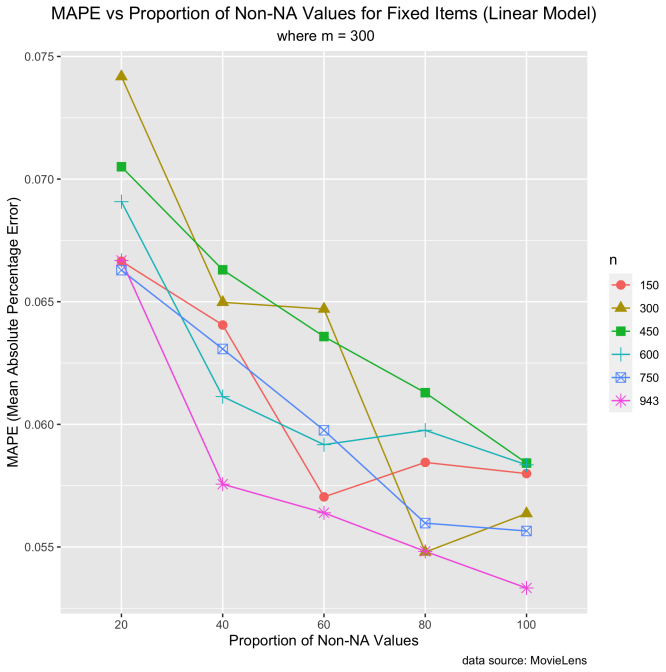

Figure 11: m-300 (mape)

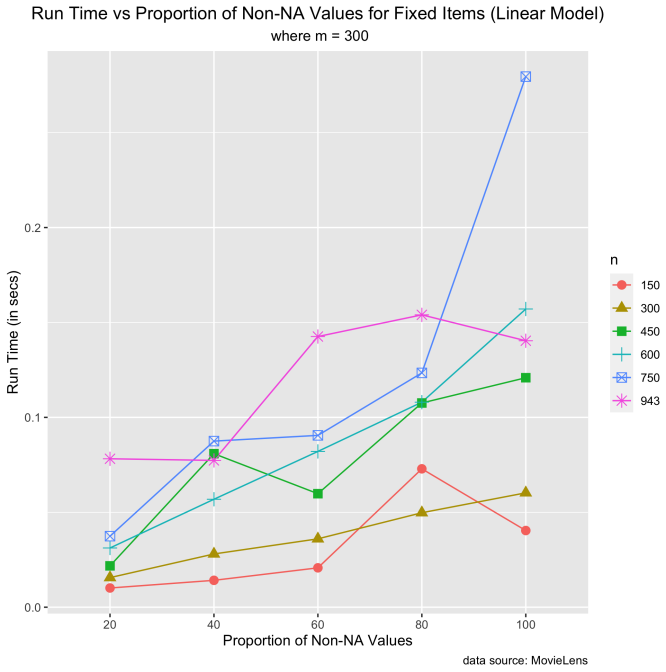

Figure 12: m-300 (run time)

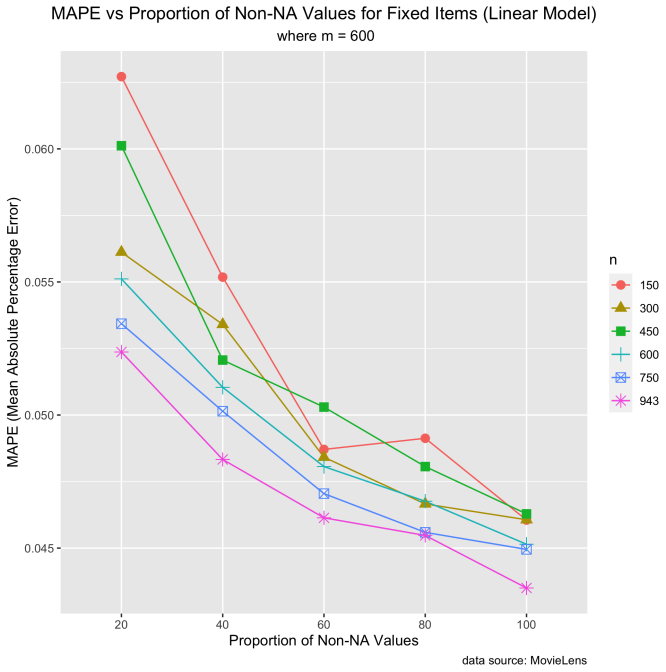

Figure 13: m-600 (mape)

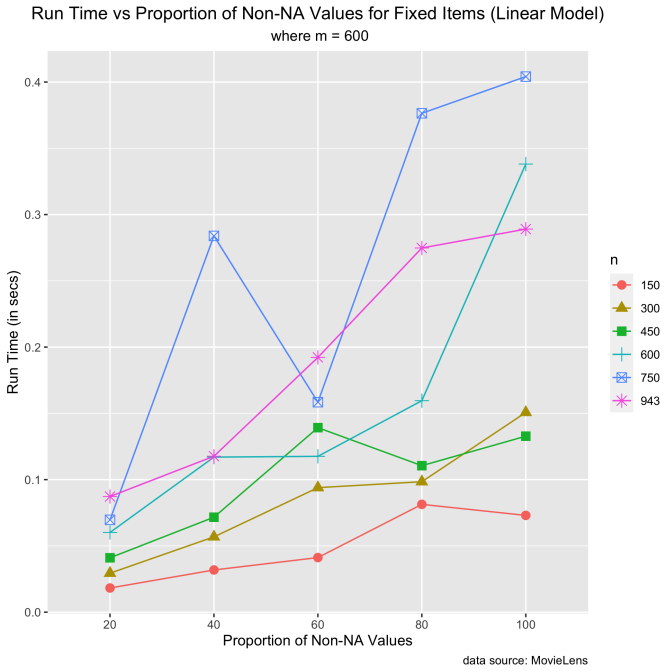

Figure 14: m-600 (run time)

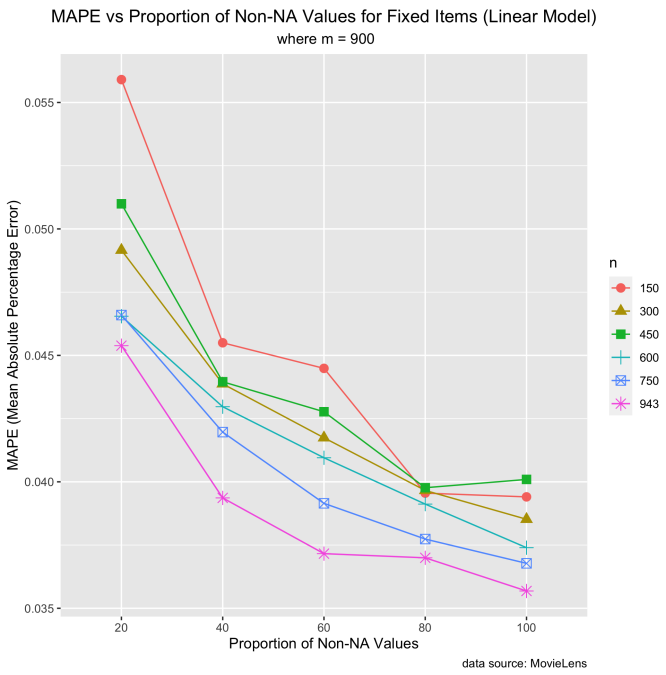

Figure 15: m-900 (mape)

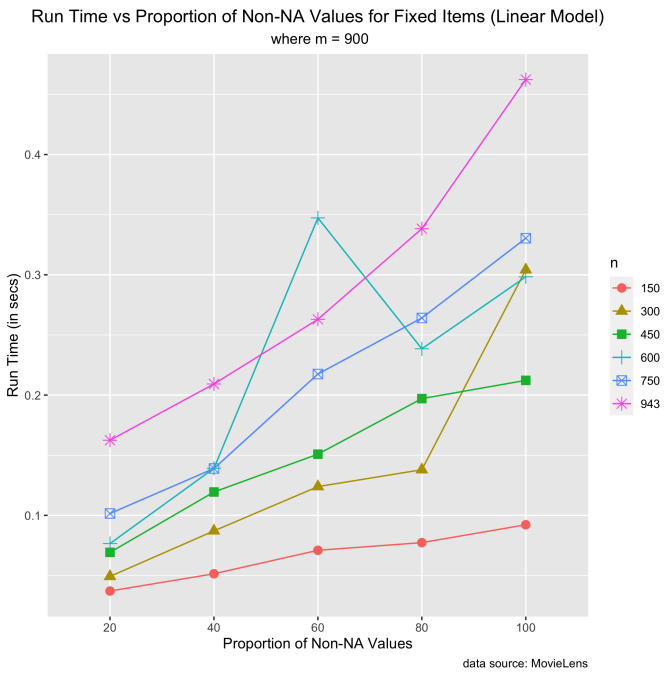

Figure 16: m-900 (run time)

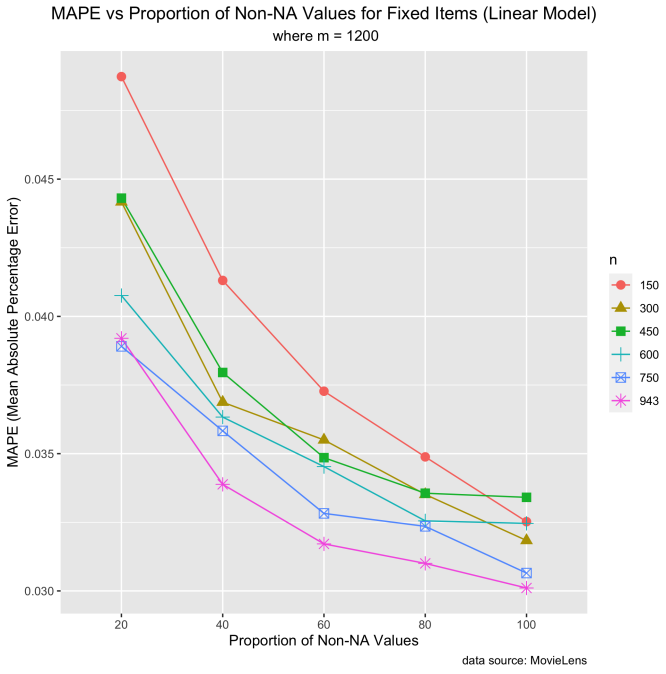

Figure 17: m-1200 (mape)

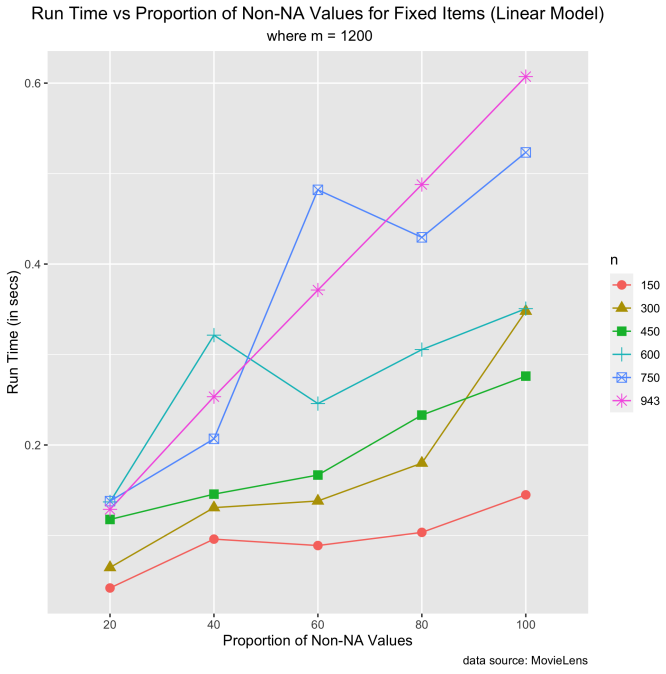

Figure 18: m-1200 (run time)

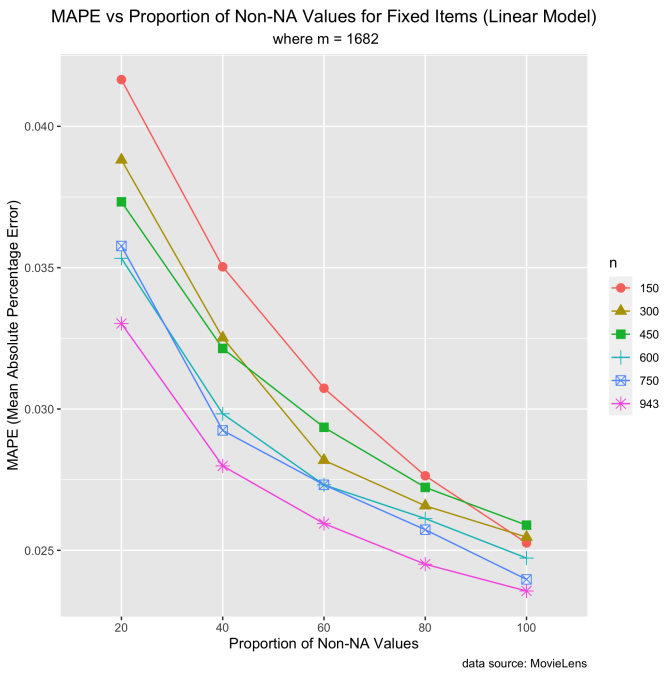

Figure 19: m-1682 (mape)

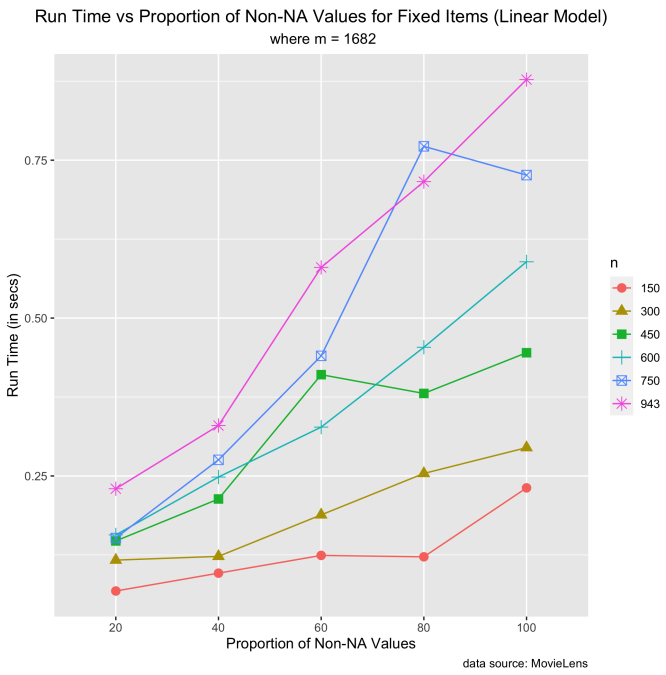

Figure 20: m-1682 (run time)

Fixed N Compare across curves:

As M increases, curves of MAPE move downward, curves of run time move upward.

Compare along curves:

As N increases, MAPE decreases and run time increases.

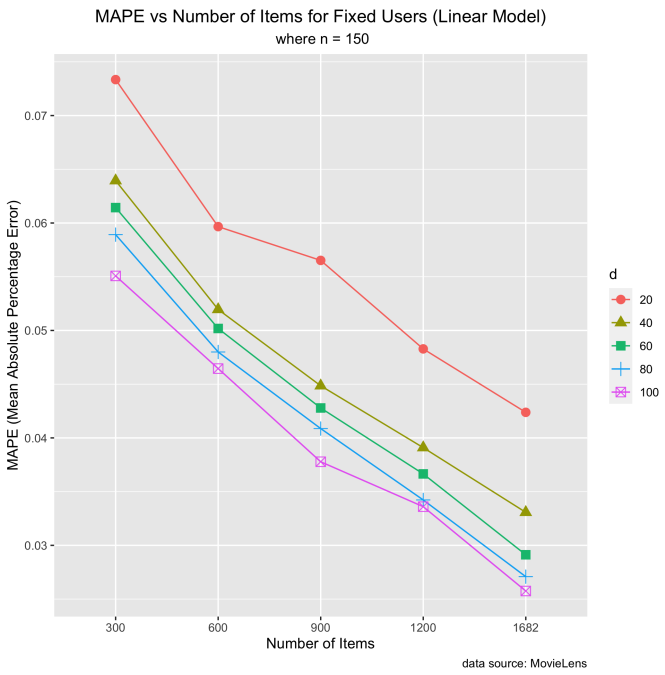

Figure 21: n-150 (mape)

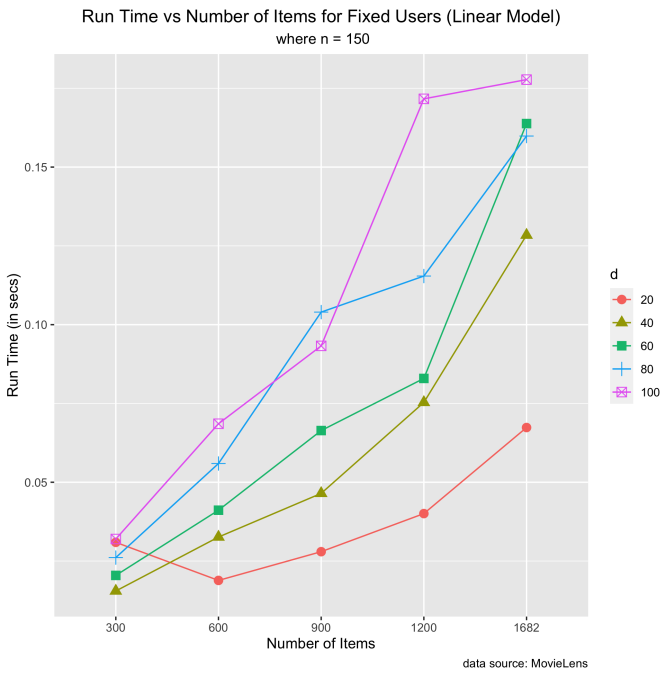

Figure 22: n-150 (run time)

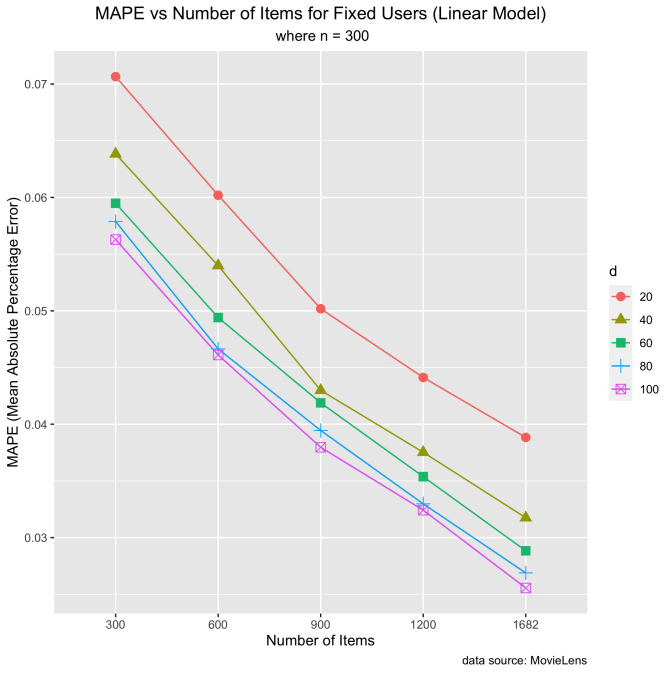

Figure 23: n-300 (mape)

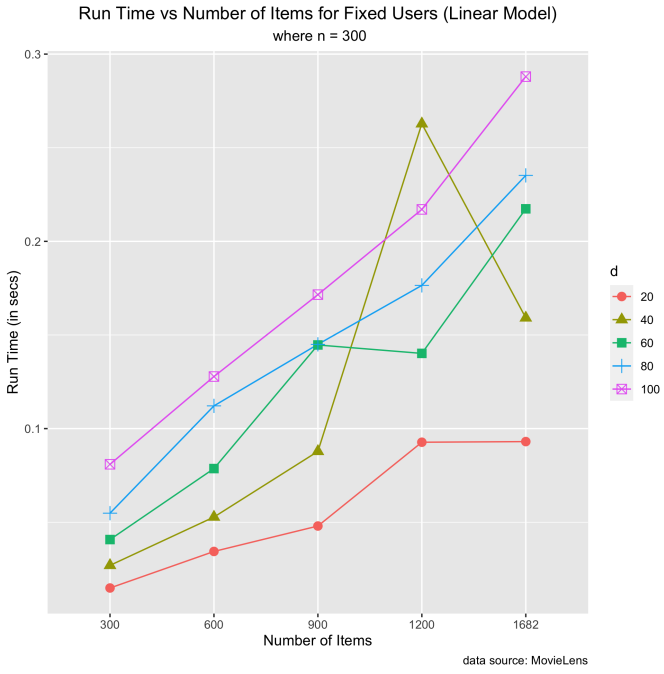

Figure 24: n-300 (run time)

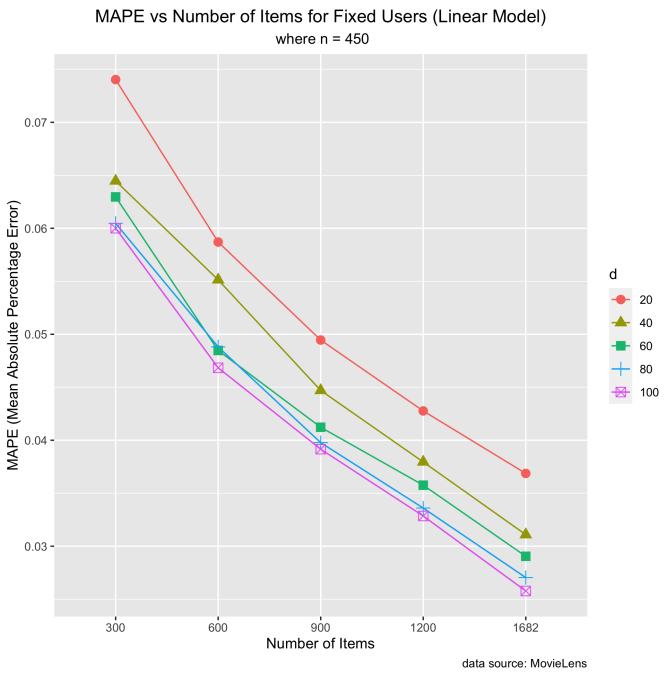

Figure 25: n-450 (mape)

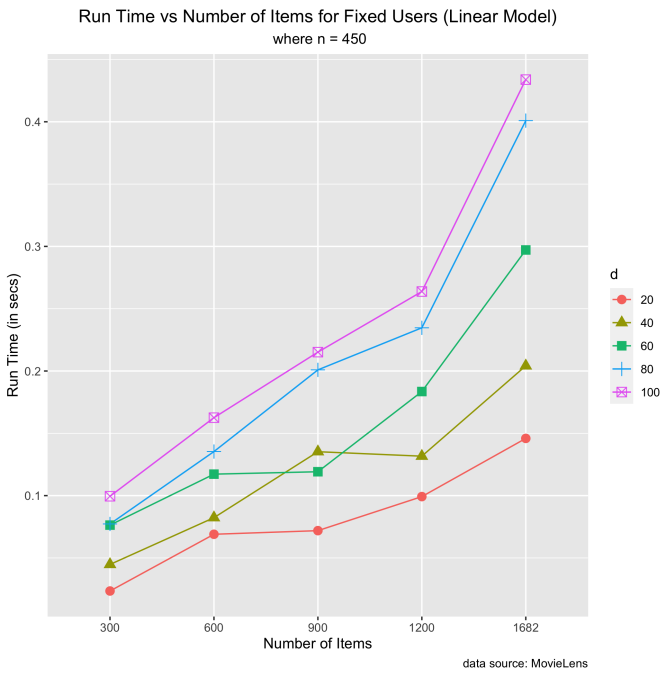

Figure 26: n-450 (run time)

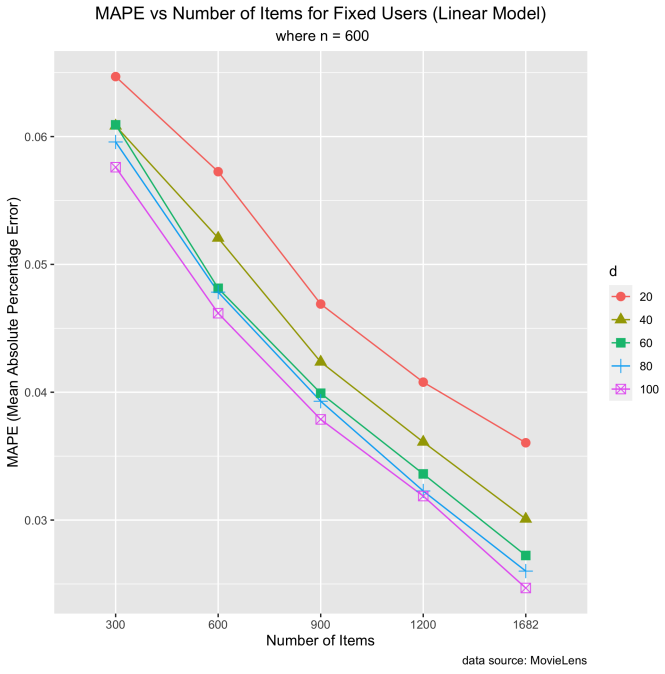

Figure 27: n-600 (mape)

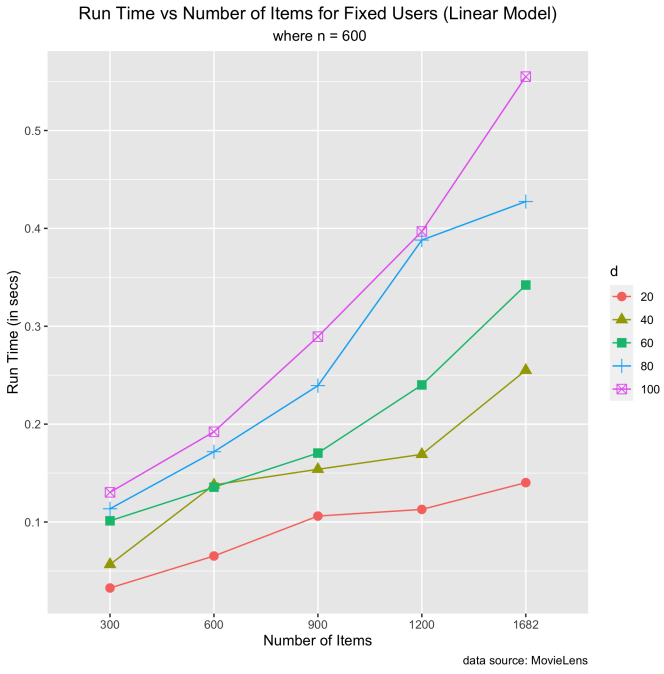

Figure 28: n-600 (run time)

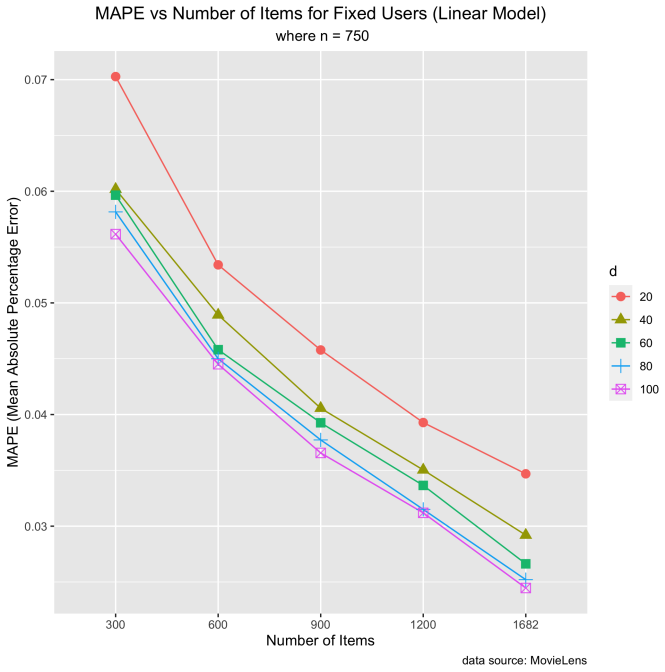

Figure 29: n-750 (mape)

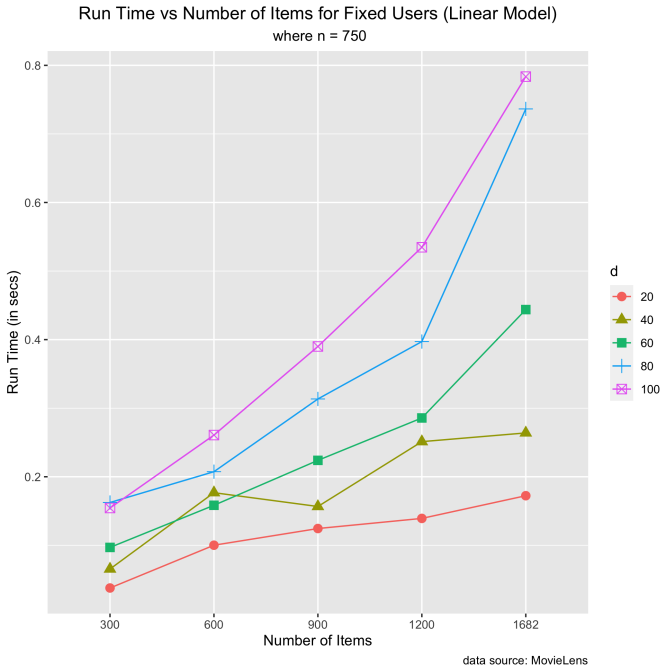

Figure 30: n-750 (run time)

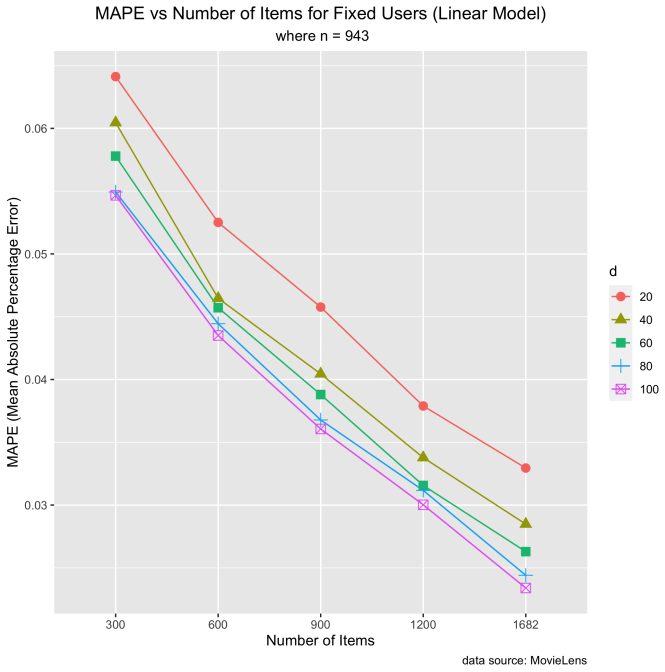

Figure 31: n-943 (mape)

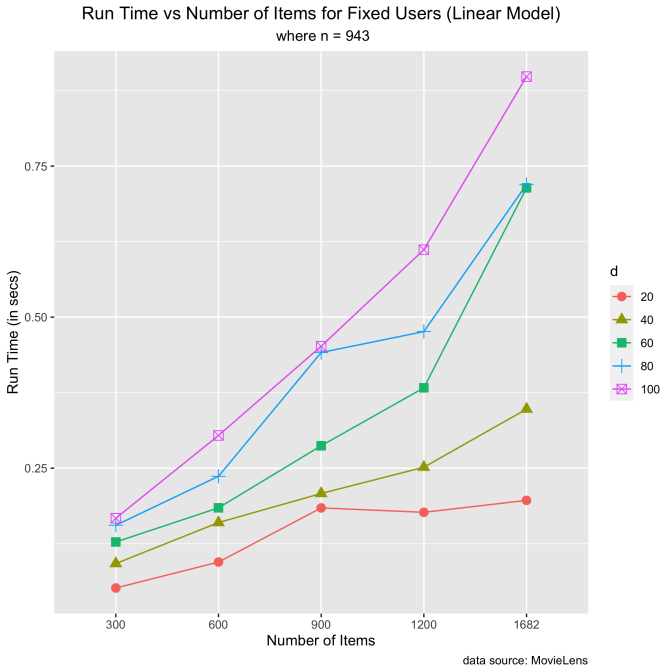

Figure 32: n-943 (run time)

3.1.2 Matrix Factorization

Fixed D

- Compare across curves: As M increases, curves of MAPE move downward, curves of run time move upward.
- Compare along curves: As N increases, MAPE decreases, but not significantly and run time increases.

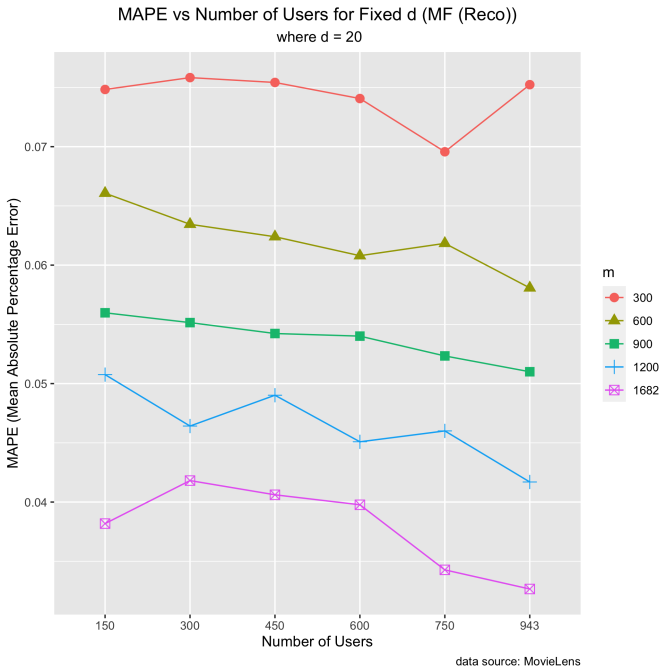

Figure 33: d-20 (mape)

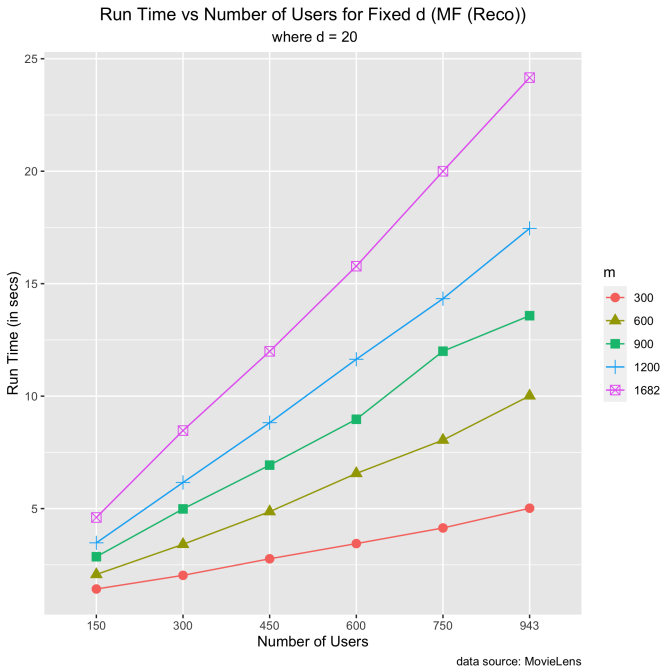

Figure 34: d-20 (run time)

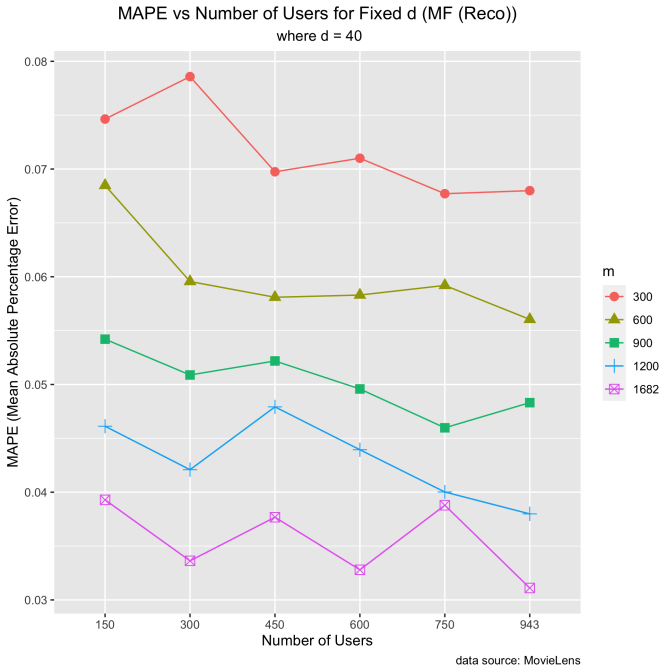

Figure 35: d-40 (mape)

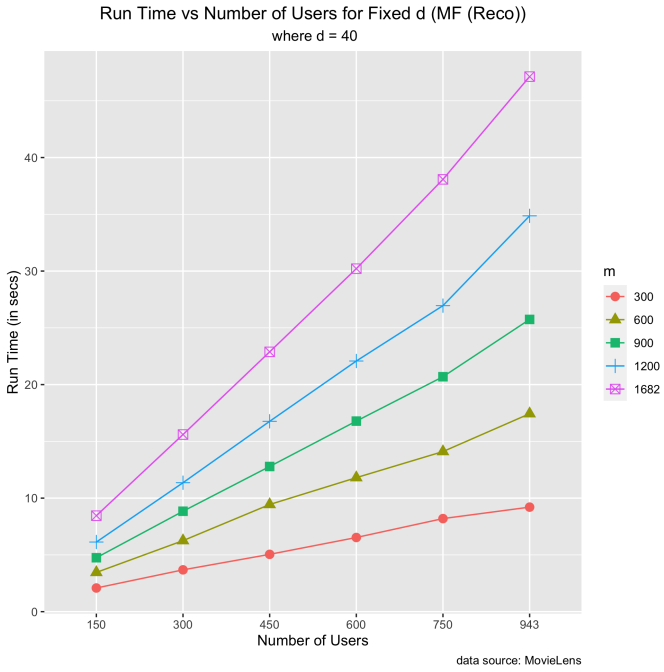

Figure 36: d-40 (run time)

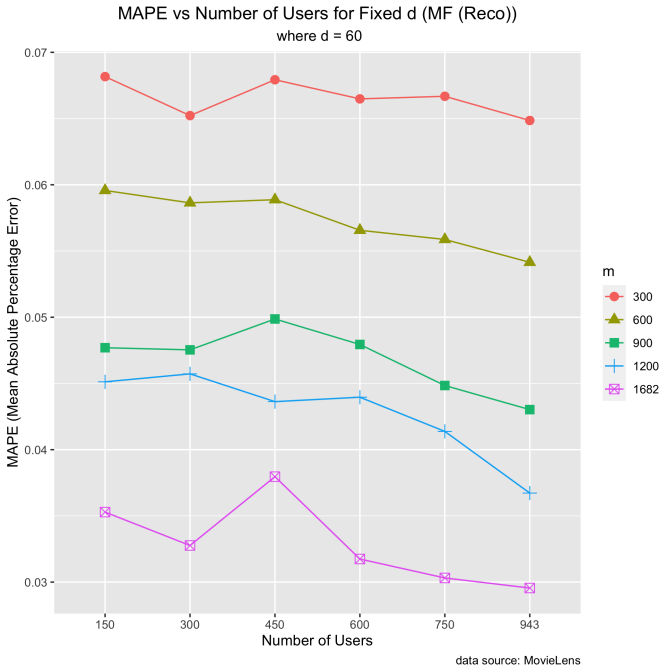

Figure 37: d-60 (mape)

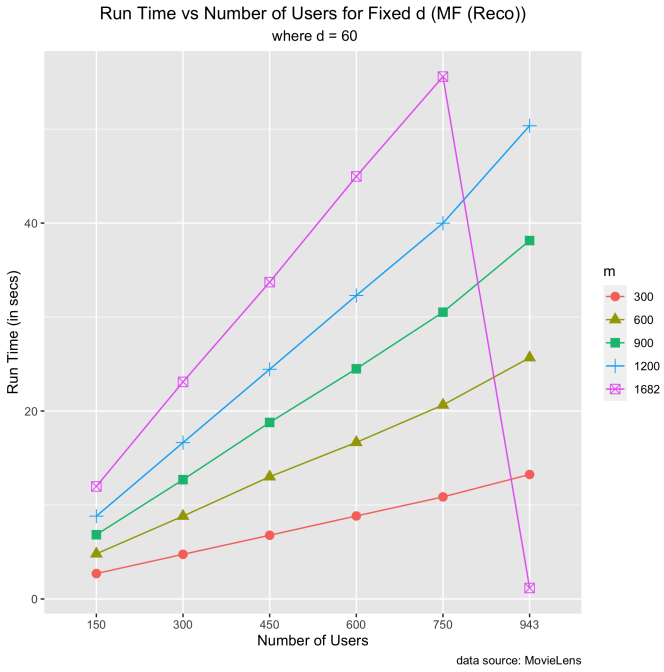

Figure 38: d-60 (run time)

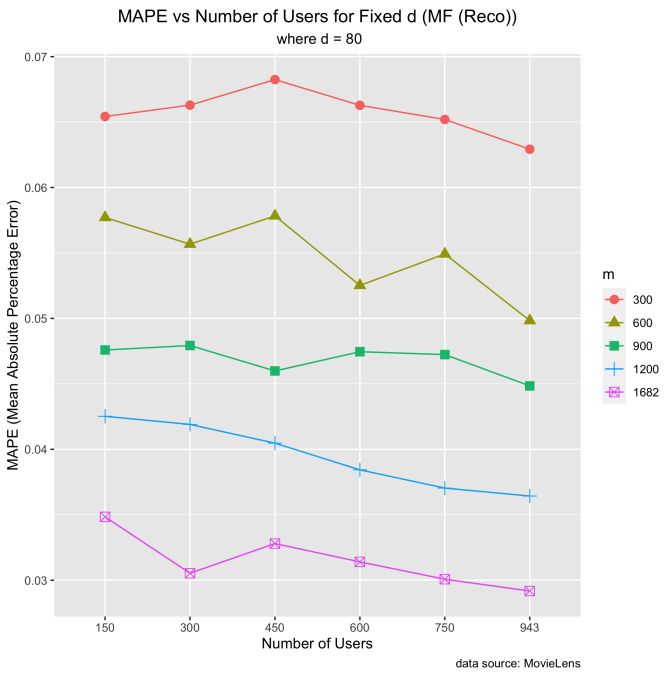

Figure 39: d-80 (mape)

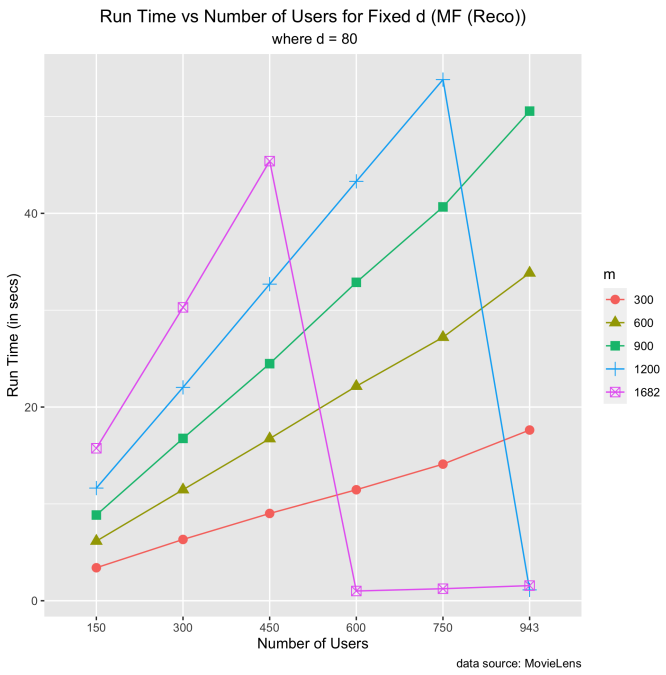

Figure 40: d-80 (run time)

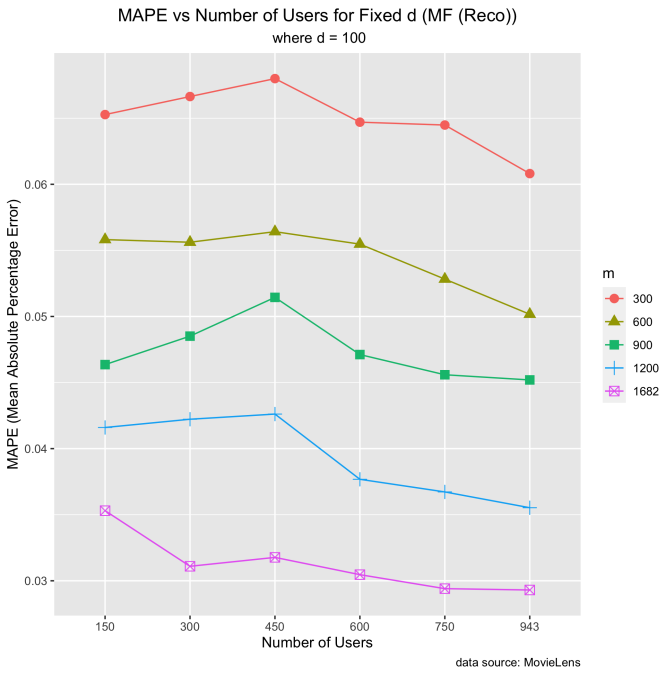

Figure 41: d-100 (mape)

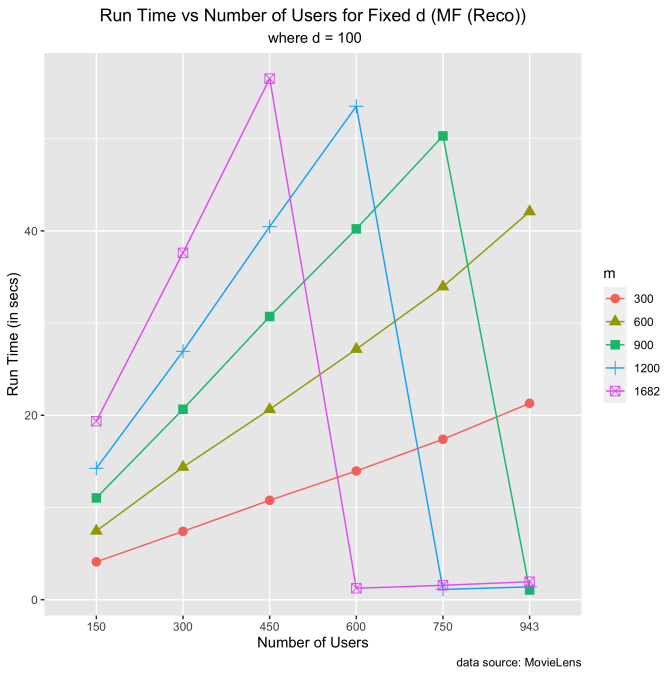

Figure 42: d-100 (run time)

Fixed M For all values of M,

- Compare across curves: when both M and N (number of users) are fixed, we can see a pattern where as D (ratio of non-NA values) increase, MAPE decreases in general.
- Compare along curves: when both M and D are fixed, the patterns between different N (number of users) values is not very distinguishable for certain N values, and details is included below. Overall, it shows a decreasing trend.

For the run time of model training, as usual, the run time increase as D or N increases.

Remarks

- when M and D (propotion of non-NA values) are fixed, and N value is not large enough, such as 150, 300, 450, the increase of N doesn't significantly and necessarily decrease the MAPE. Hence, we can see even though the sample size is important for increase the accuracy, but a small increase of sample size may not effectively increase the accuracy until a large sample comes in.

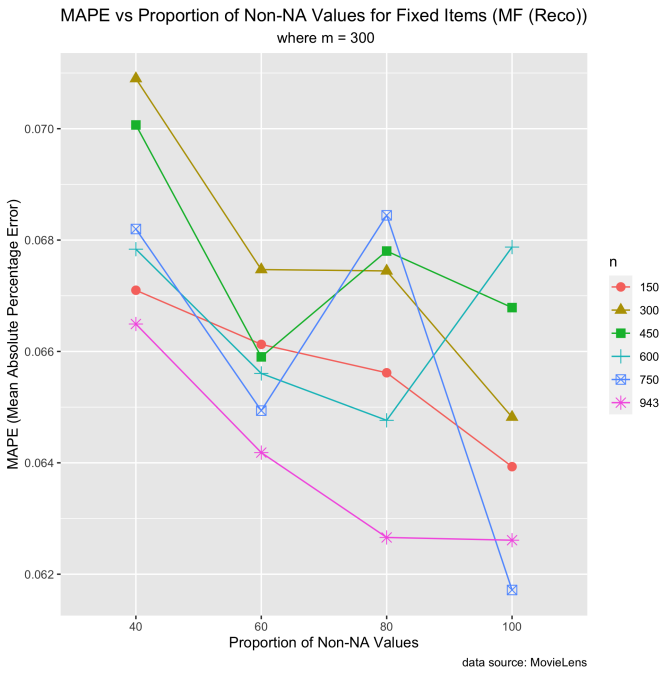

Figure 43: m-300 (mape)

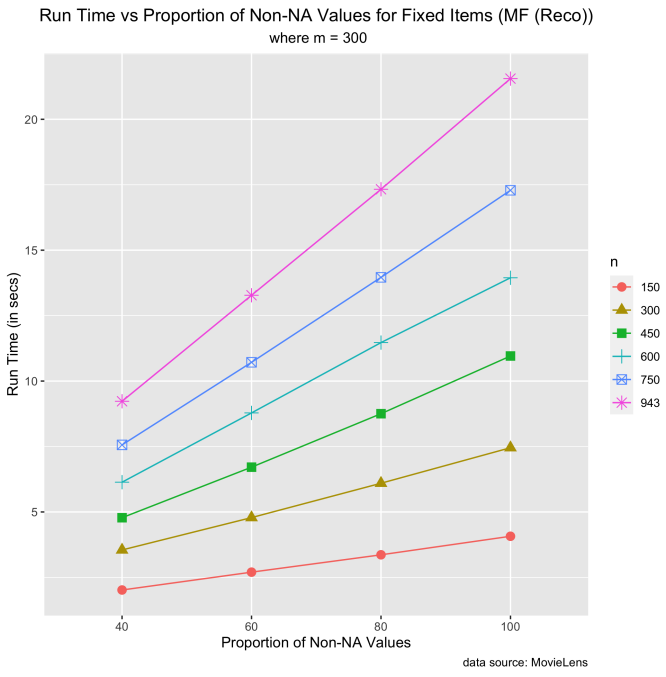

Figure 44: m-300 (run time)

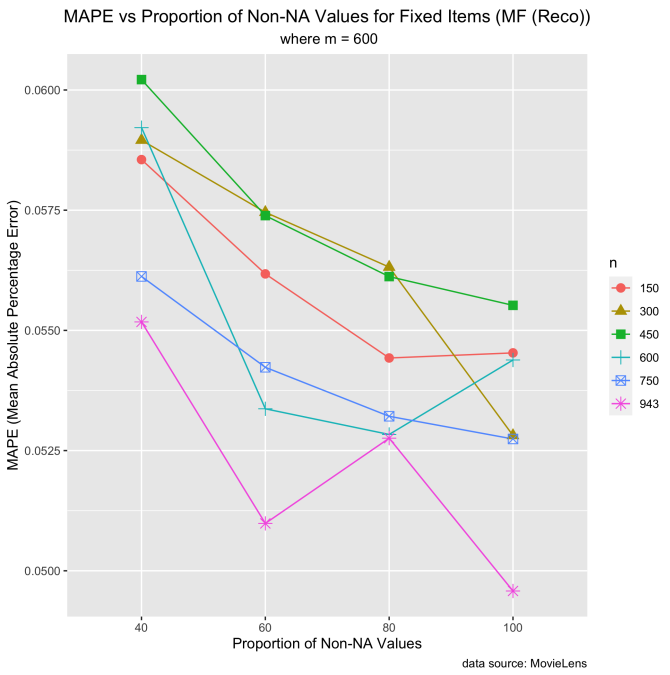

Figure 45: m-600 (mape)

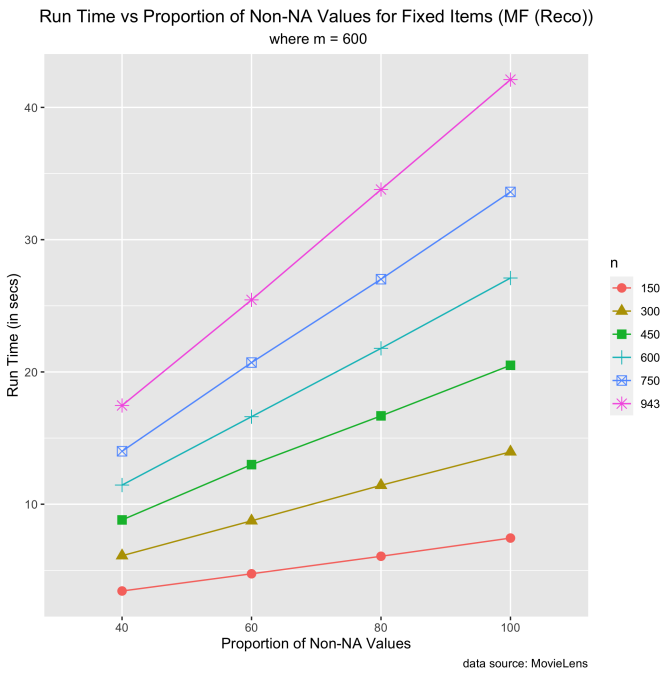

Figure 46: m-600 (run time)

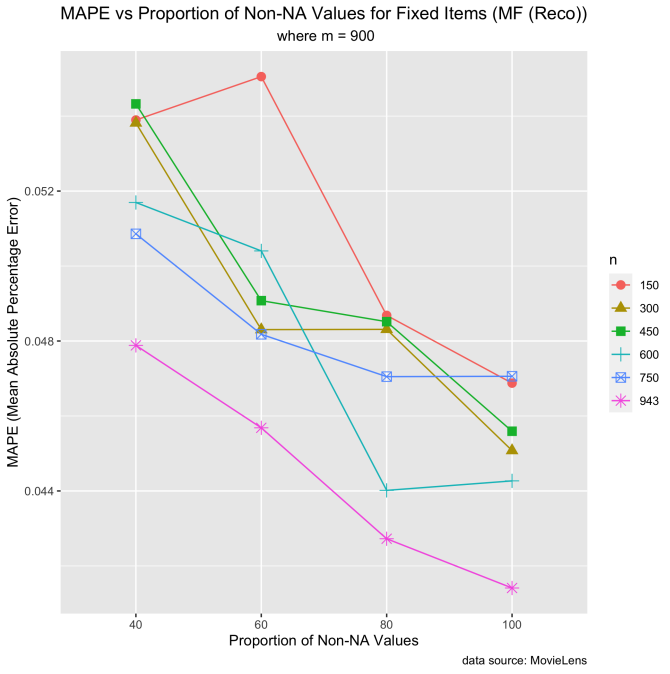

Figure 47: m-900 (mape)

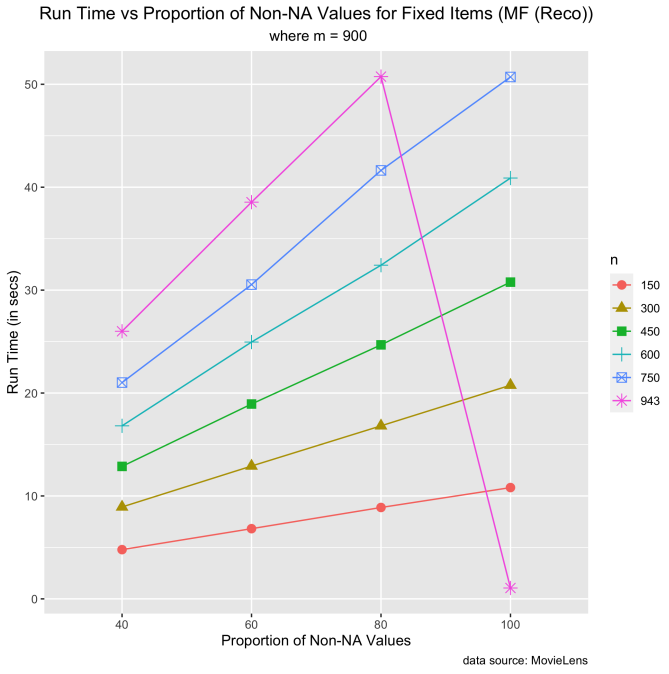

Figure 48: m-900 (run time)

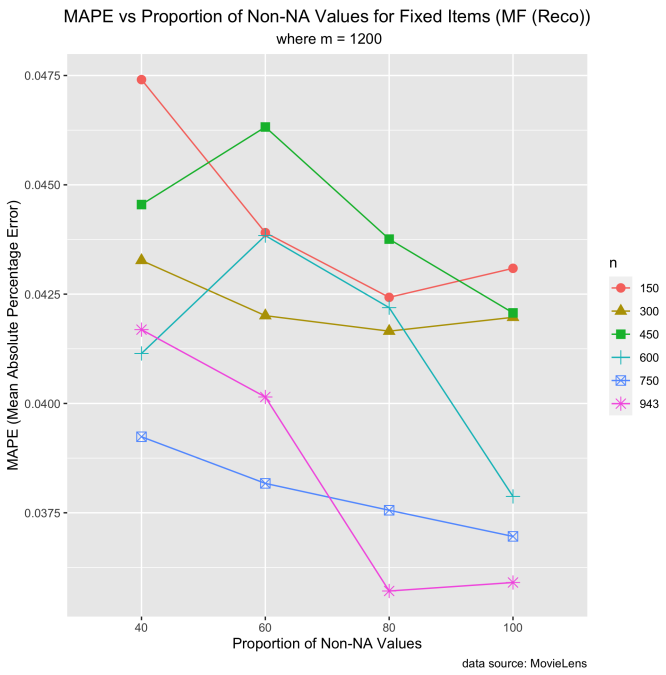

Figure 49: m-1200 (mape)

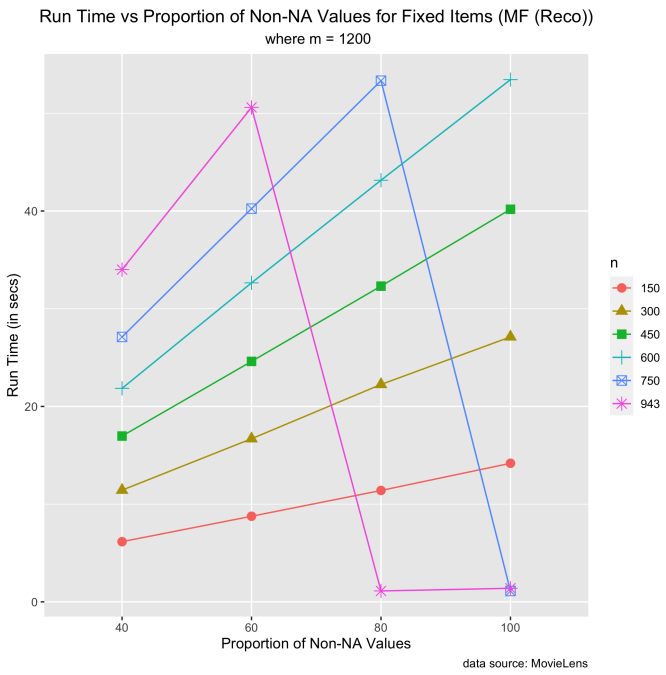

Figure 50: m-1200 (run time)

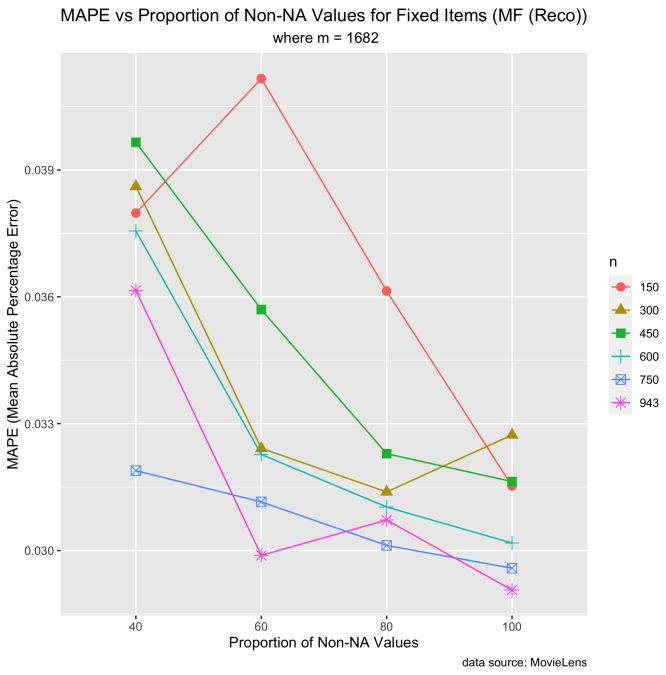

Figure 51: m-1682 (mape)

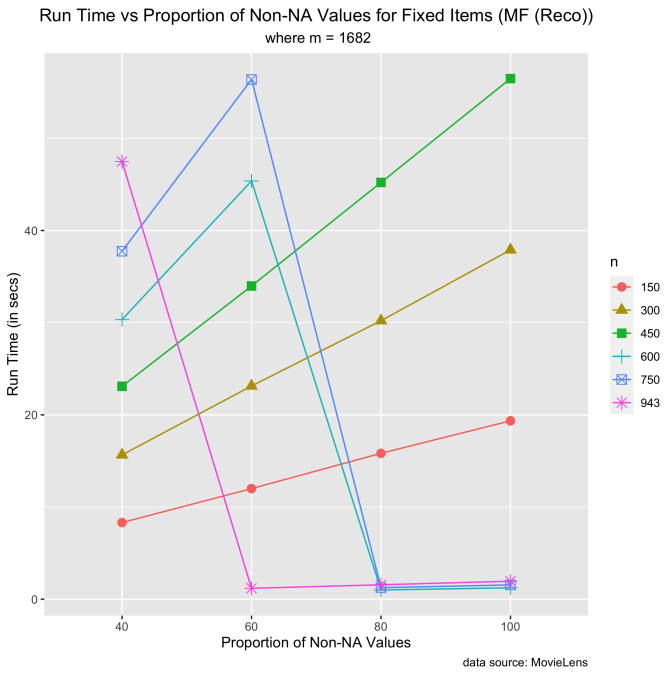

Figure 52: m-1682 (run time)

Fixed N

- Compare across curves: As d increases, curves of MAPE move downward, curves of run time move upward.
- Compare along curves: As M increases, MAPE decreases and run time increases.

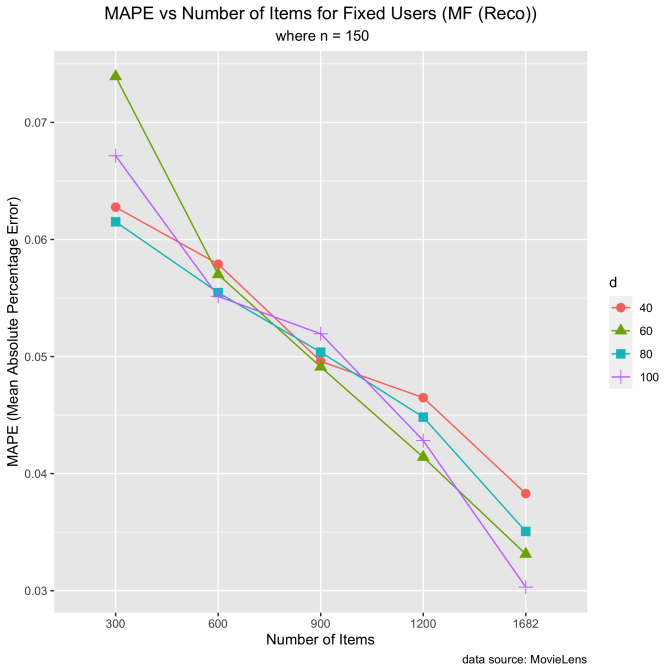

Figure 53: n-150 (mape)

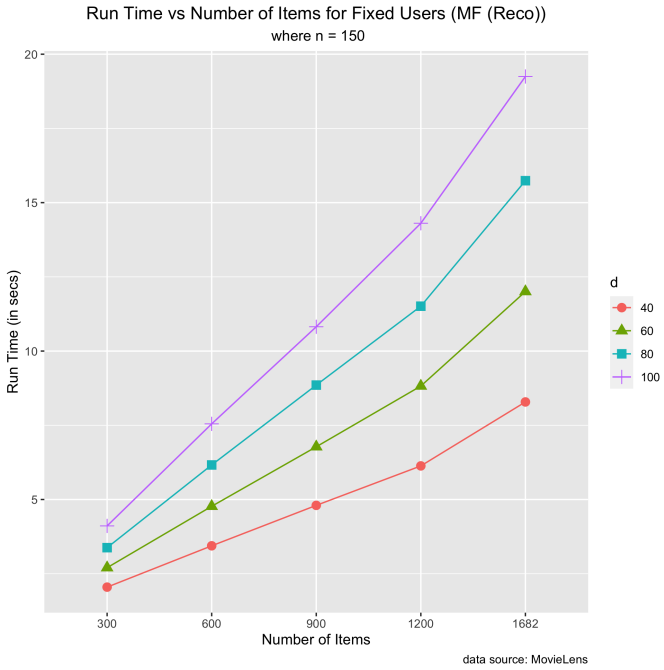

Figure 54: n-150 (run time)

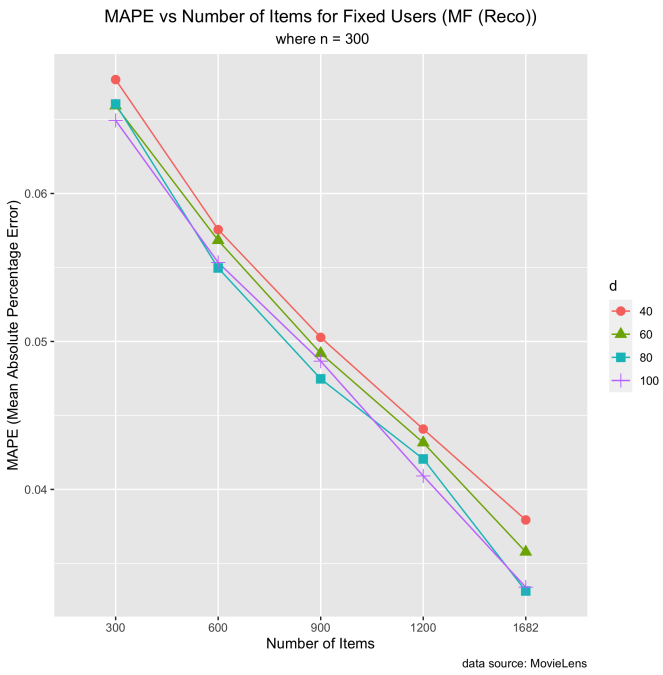

Figure 55: n-300 (mape)

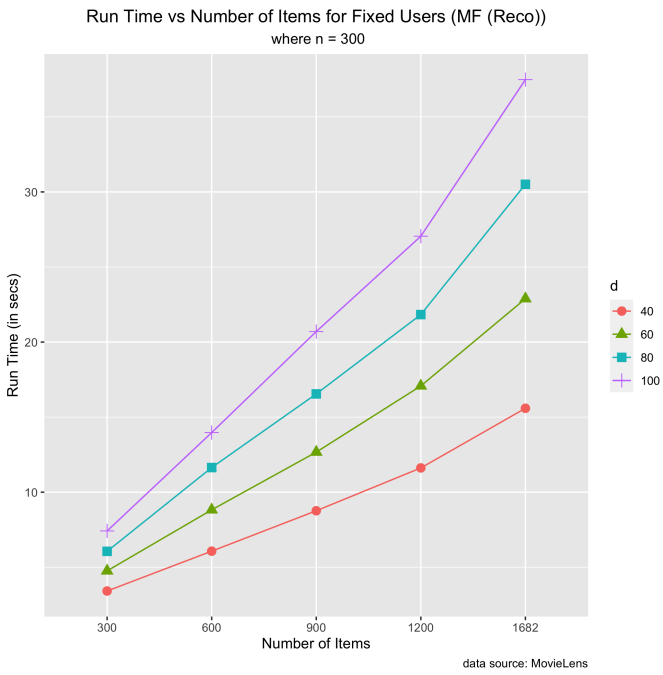

Figure 56: n-300 (run time)

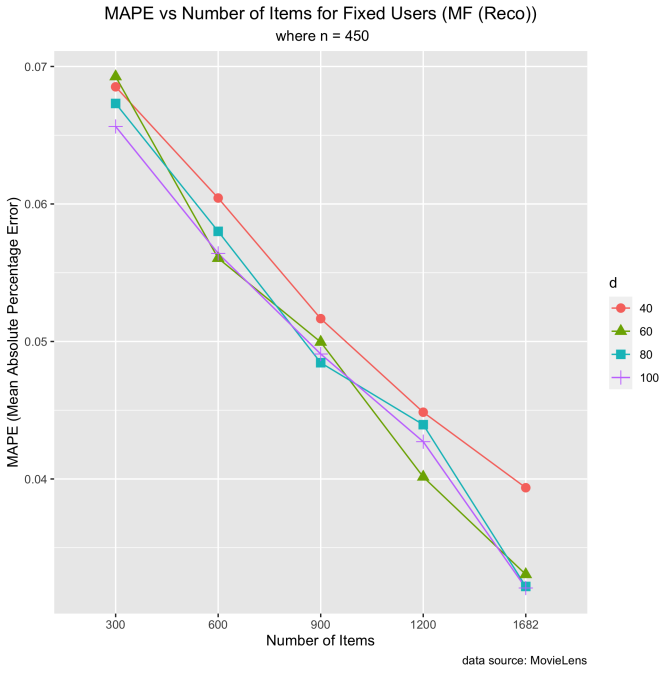

Figure 57: n-450 (mape)

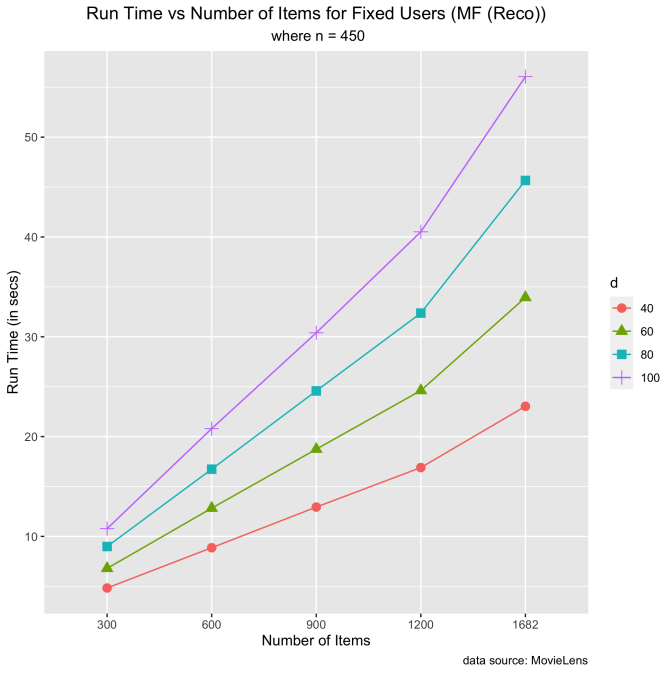

Figure 58: n-450 (run time)

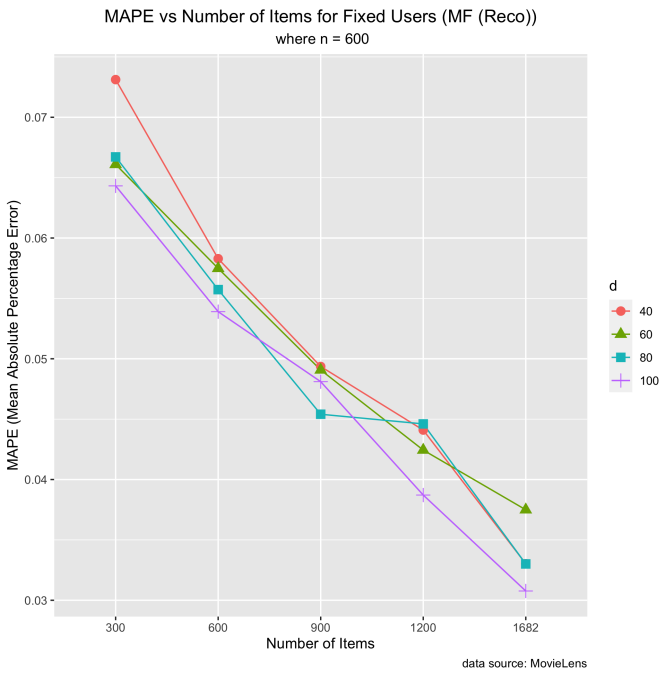

Figure 59: n-600 (mape)

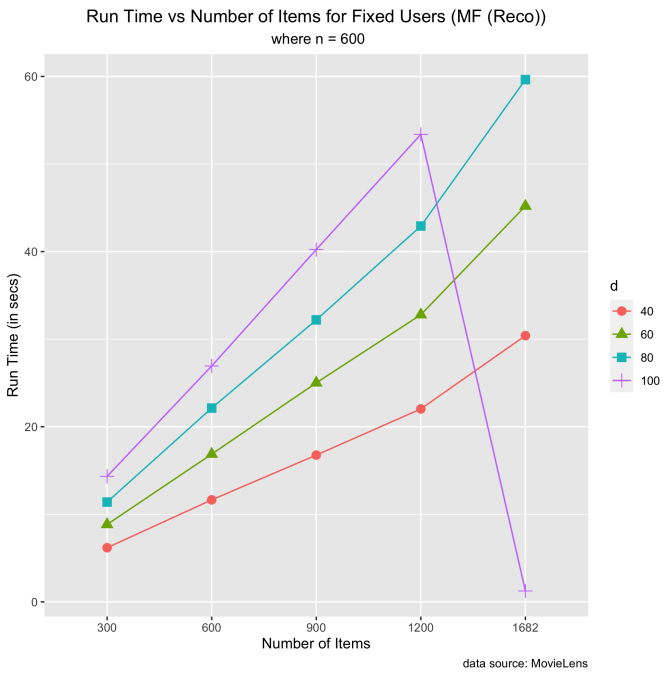

Figure 60: n-600 (run time)

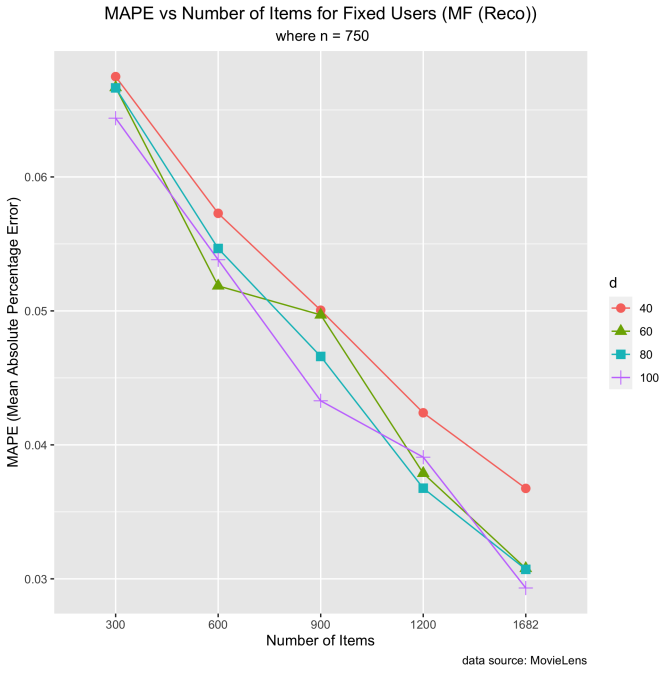

Figure 61: n-750 (mape)

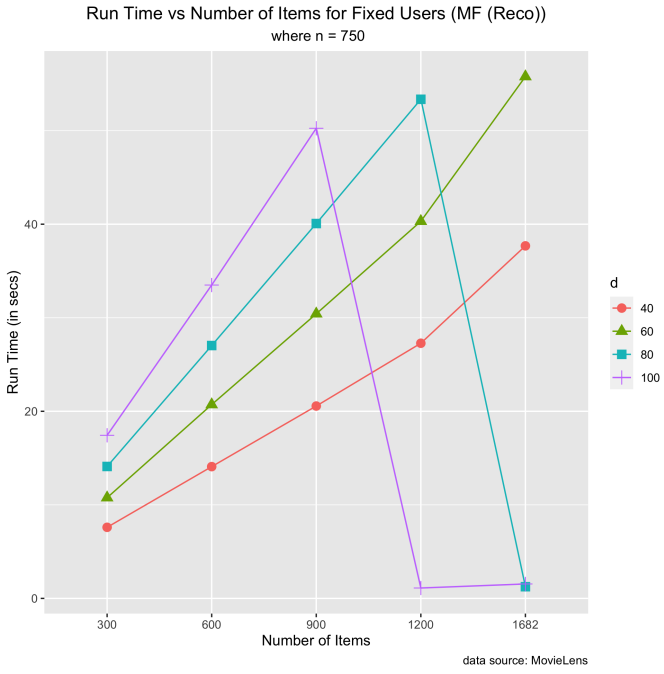

Figure 62: n-750 (run time)

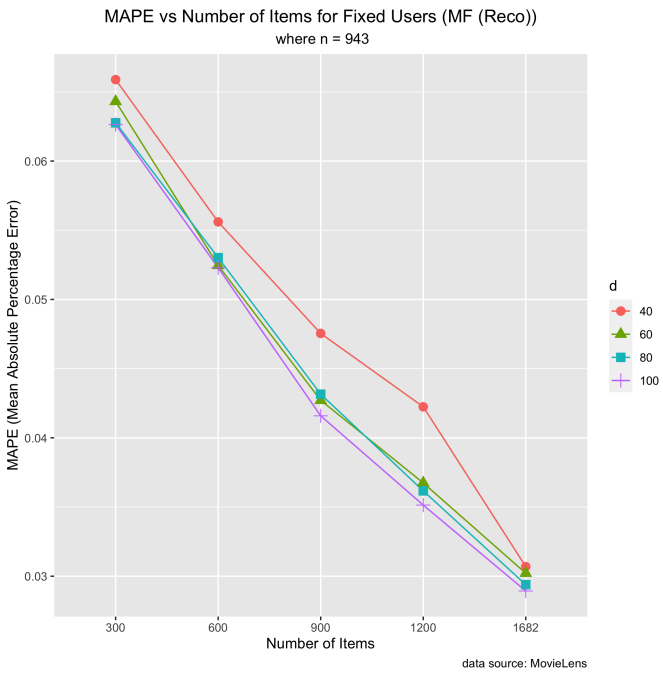

Figure 63: n-943 (mape)

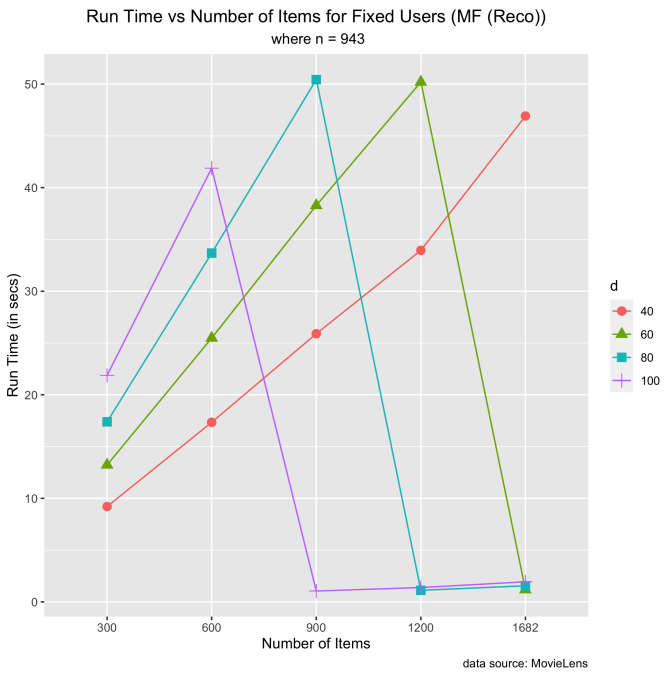

Figure 64: n-943 (run time)

3.2 InstEval

3.2.1 Linear Model

Fixed D

- Compare across curves: As m increases, curves of MAPE move downward, curves of run time move upward.
- Compare along curves: As N increases, MAPE decreases and run time increases.

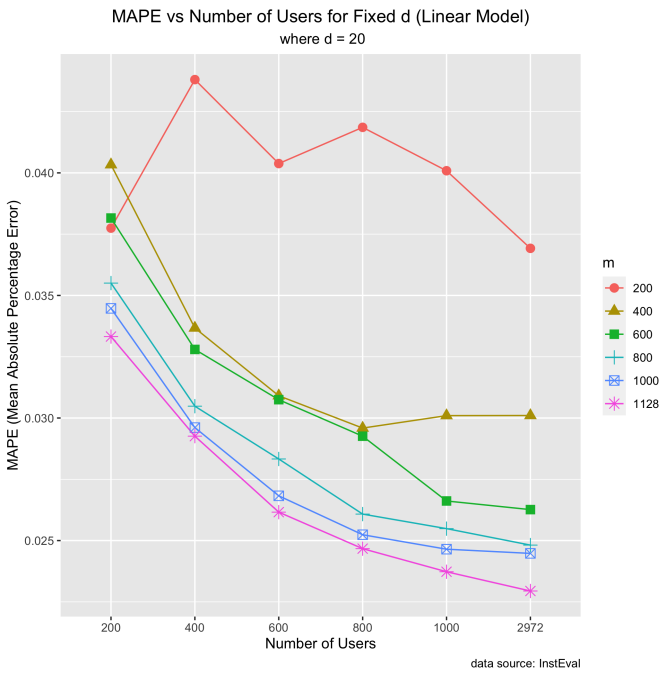

Figure 65: d-20 (mape)

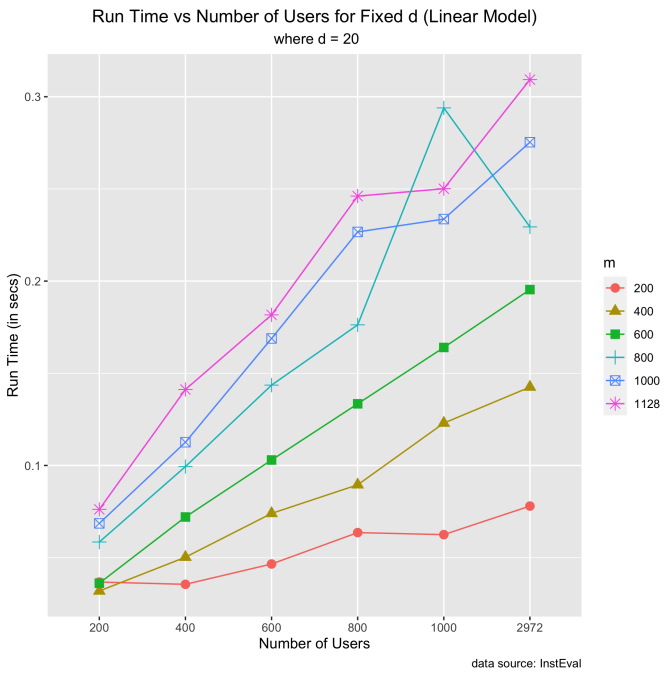

Figure 66: d-20 (run time)

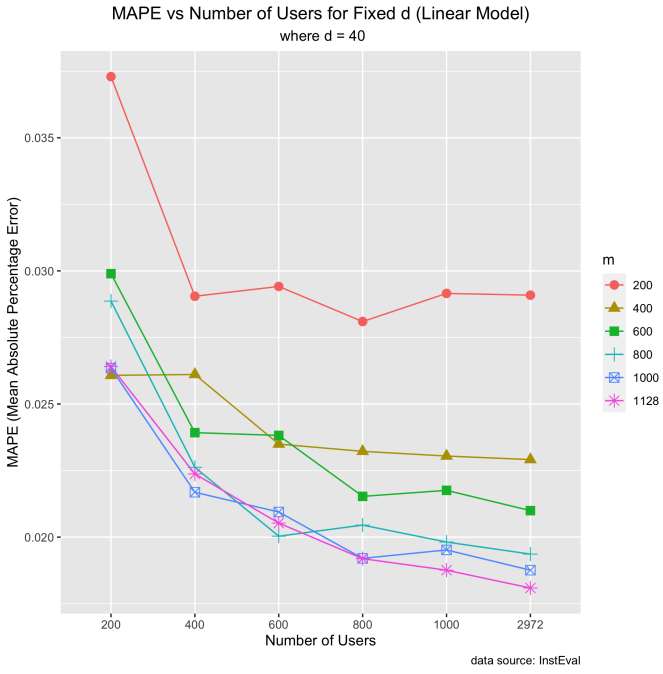

Figure 67: d-40 (mape)

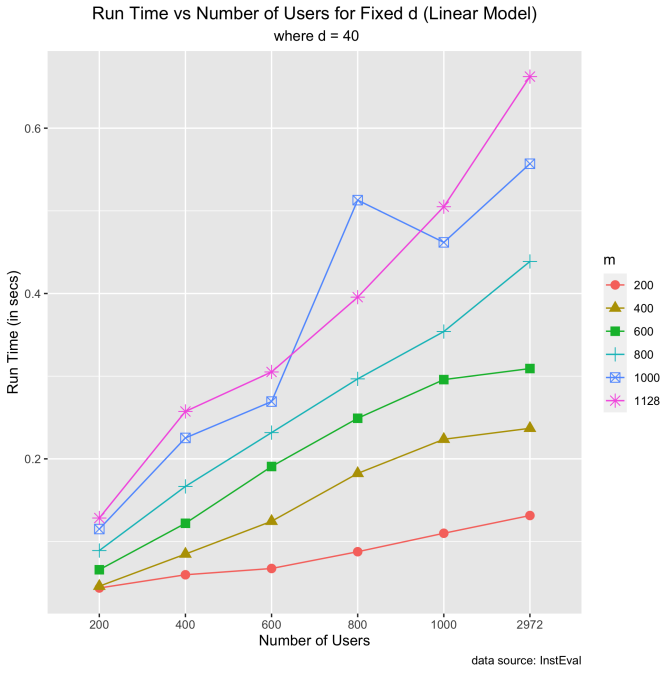

Figure 68: d-40 (run time)

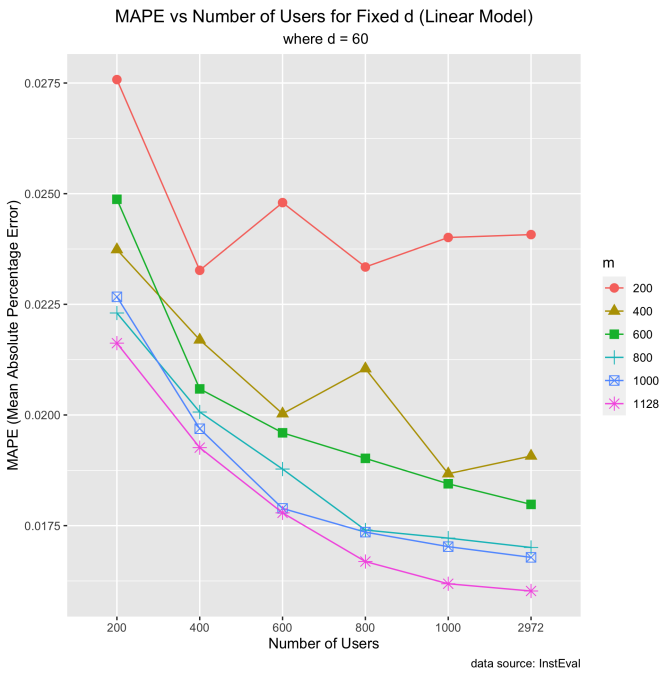

Figure 69: d-60 (mape)

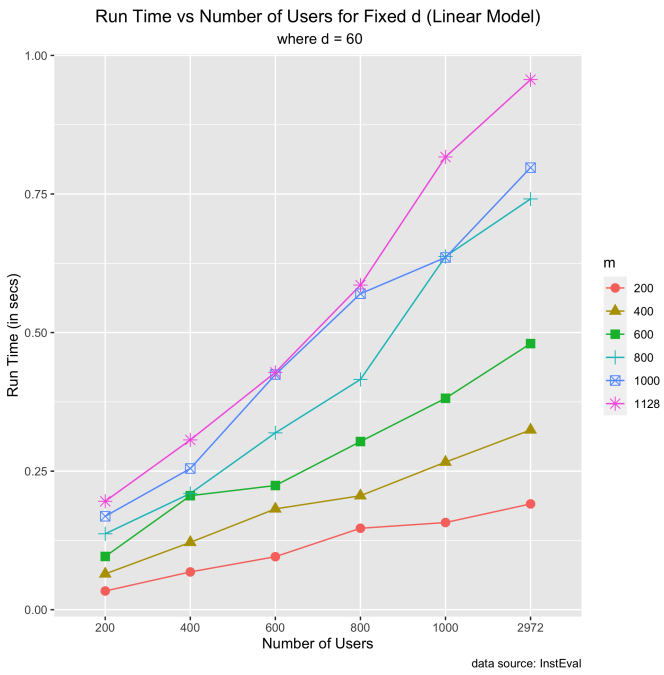

Figure 70: d-60 (run time)

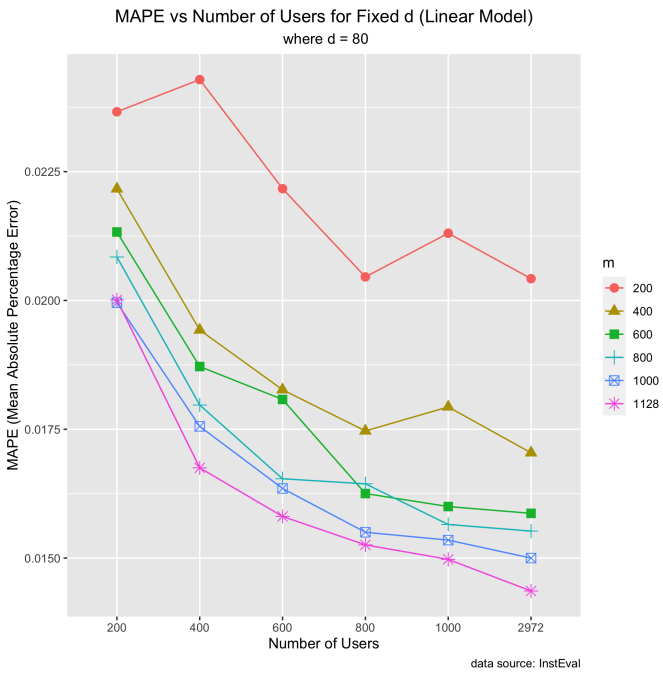

Figure 71: d-80 (mape)

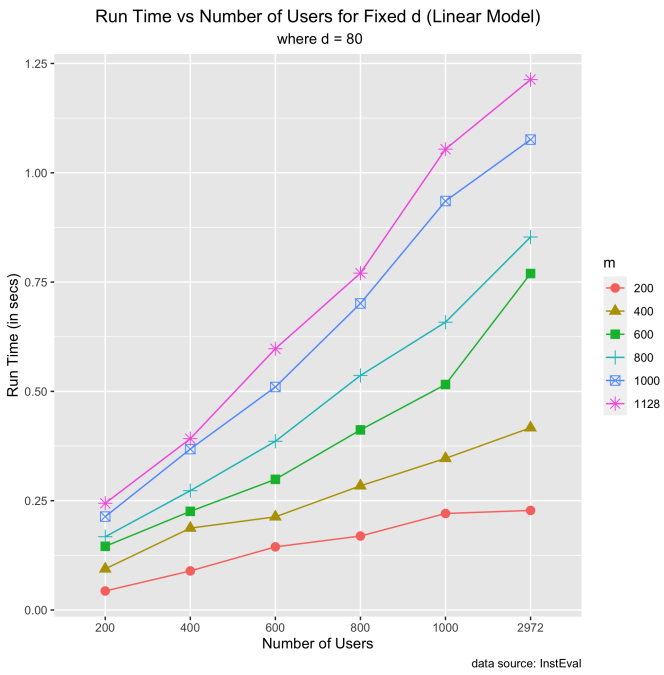

Figure 72: d-80 (run time)

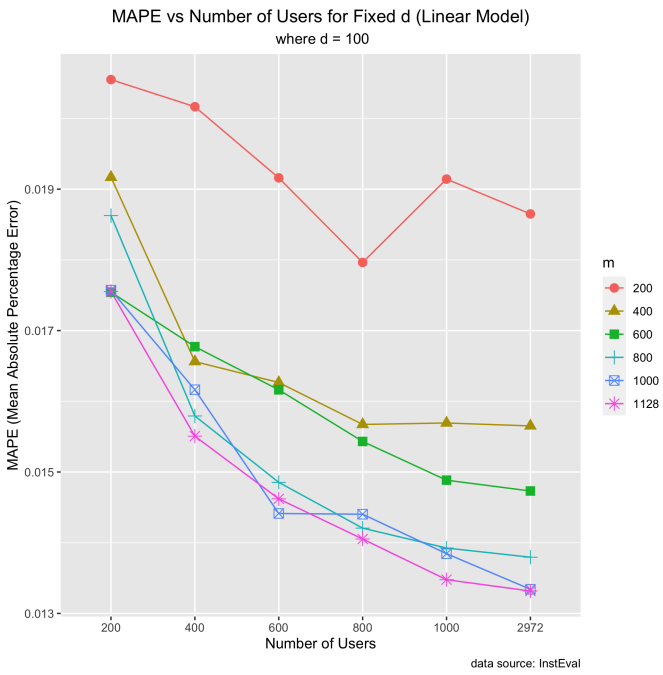

Figure 73: d-100 (mape)

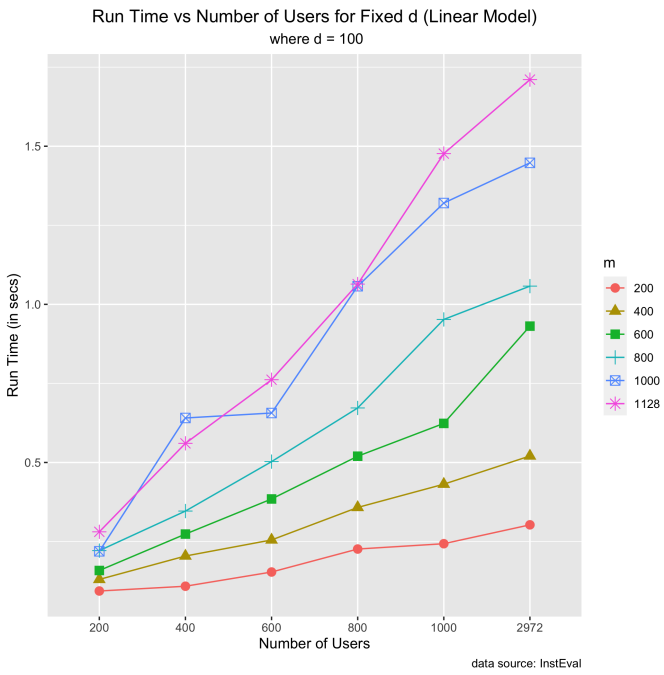

Figure 74: d-100 (run time)

Fixed M

- Compare across curves: As n increases, curves of MAPE move downward, curves of run time move upward.
- Compare along curves: As d increases, MAPE decreases and run time increases.

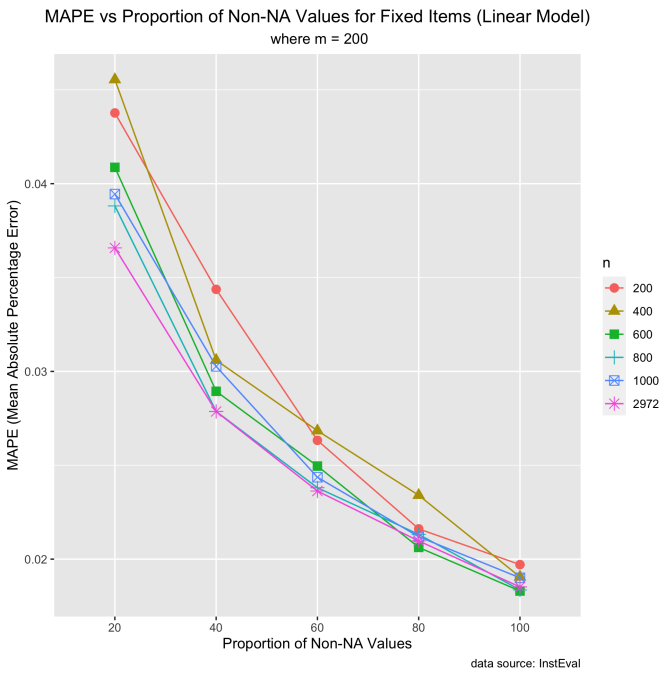

Figure 75: m-200 (mape)

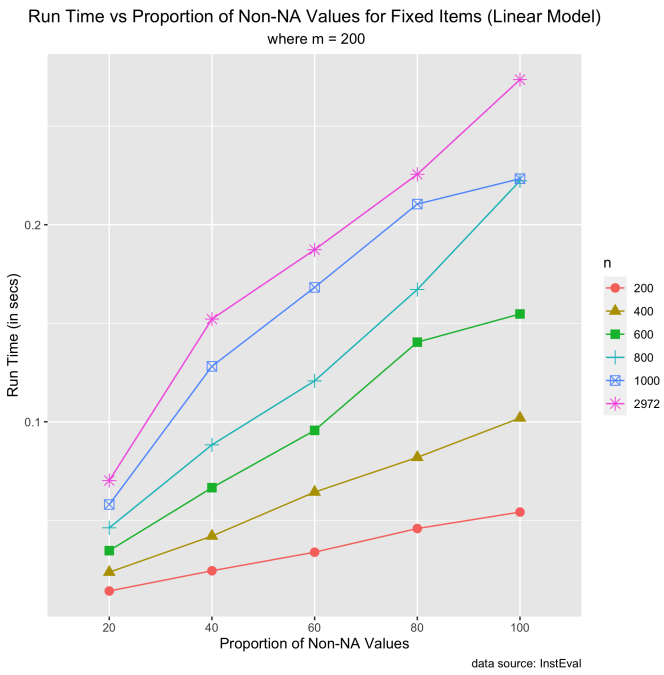

Figure 76: m-200 (run time)

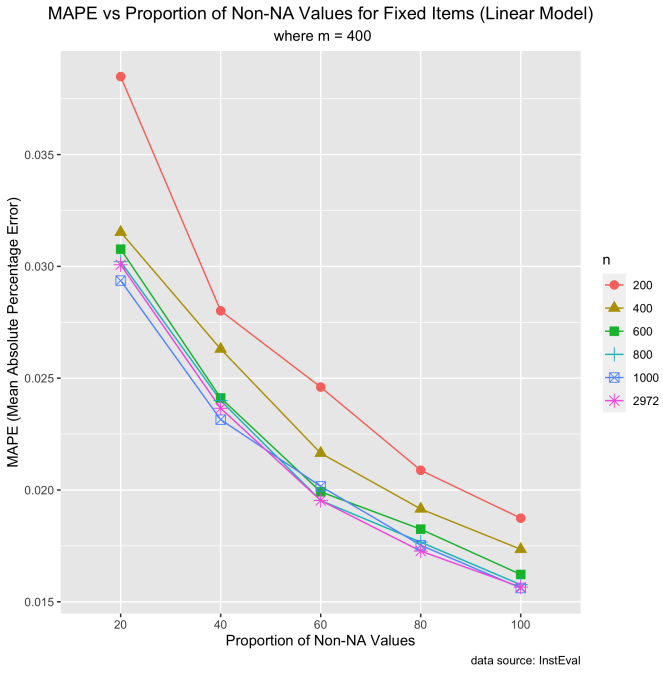

Figure 77: m-400 (mape)

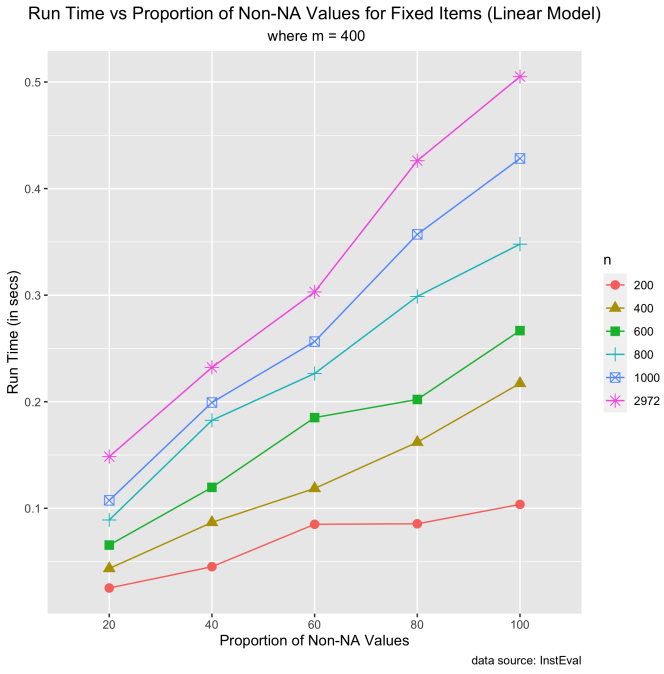

Figure 78: m-400 (run time)

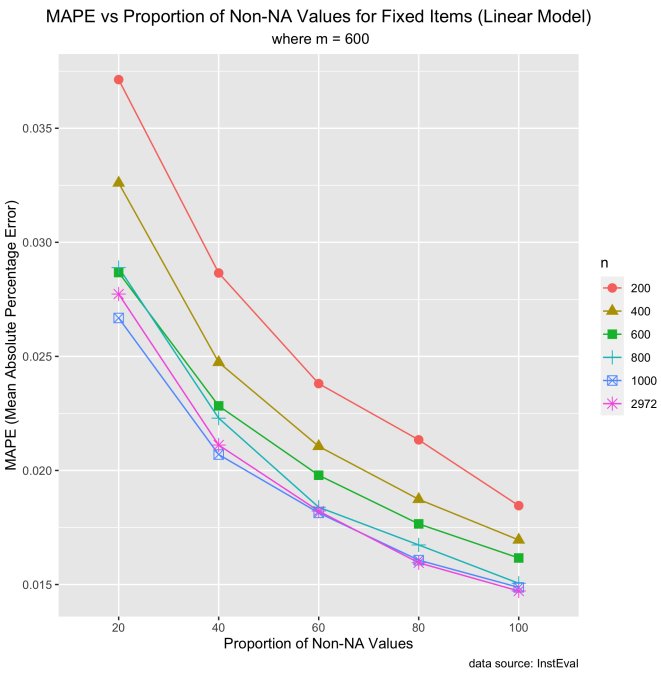

Figure 79: m-600 (mape)

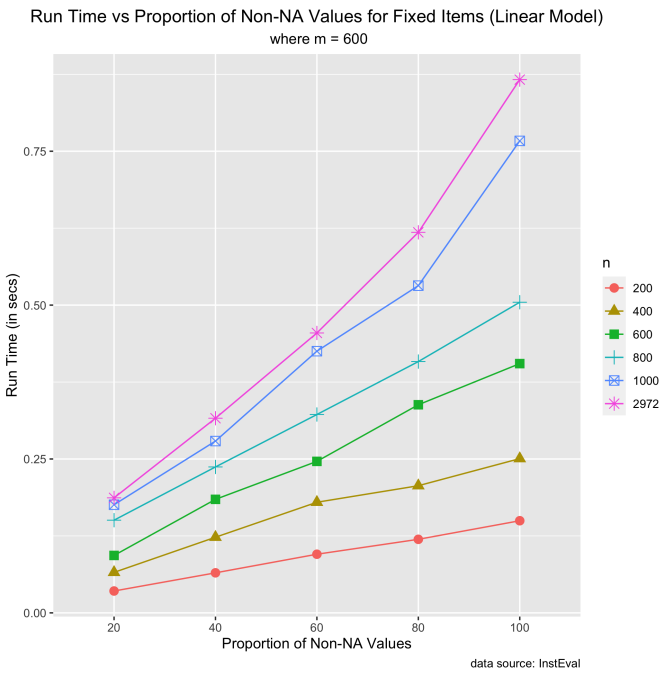

Figure 80: m-600 (run time)

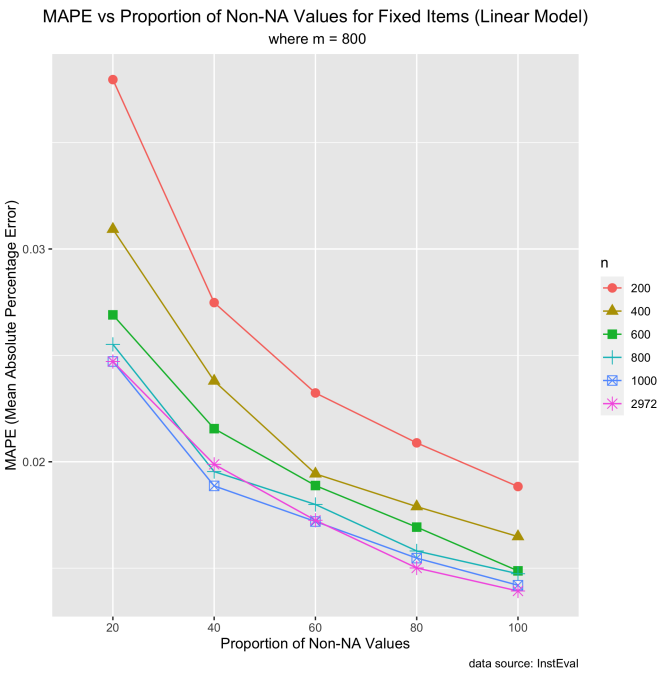

Figure 81: m-800 (mape)

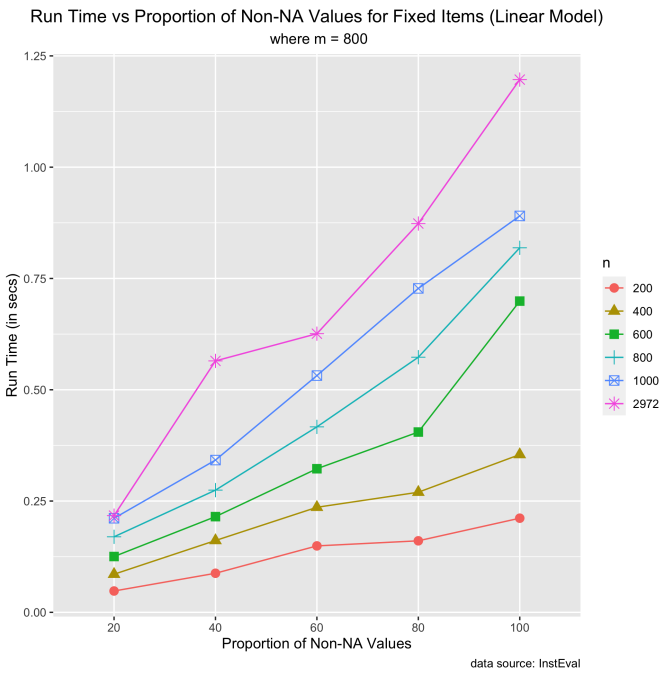

Figure 82: m-800 (run time)

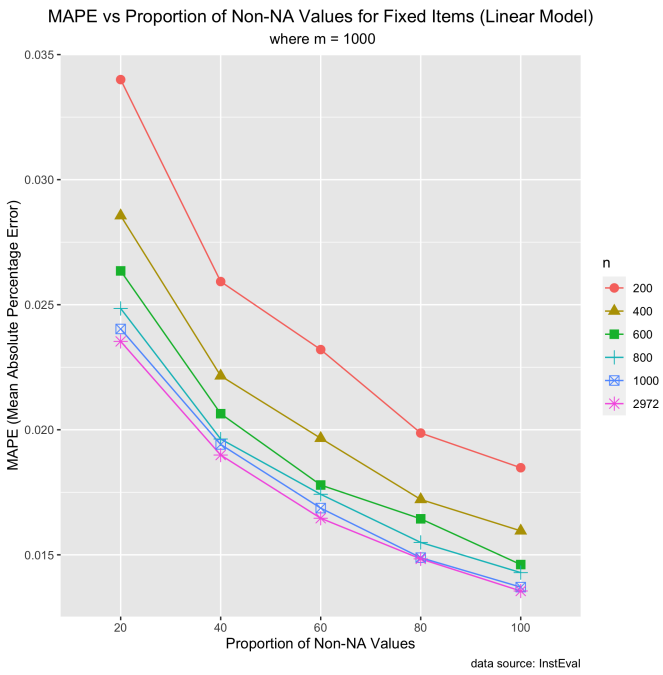

Figure 83: m-1000 (mape)

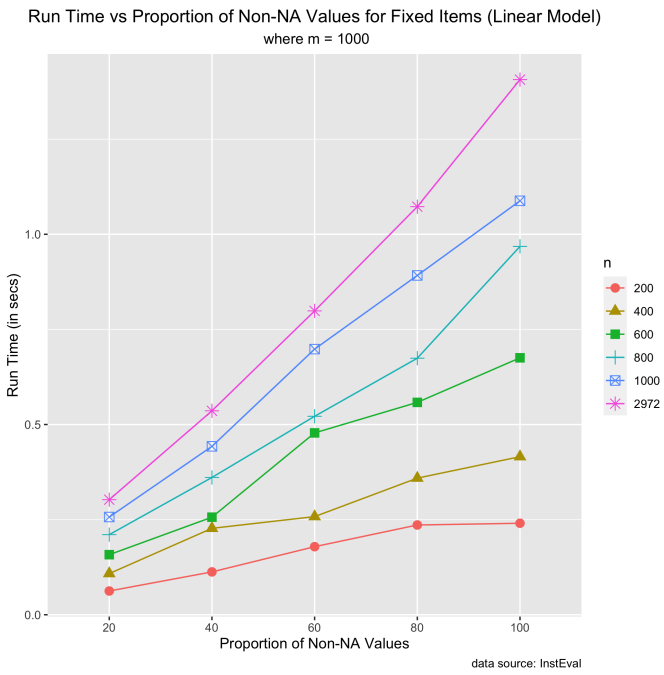

Figure 84: m-1000 (run time)

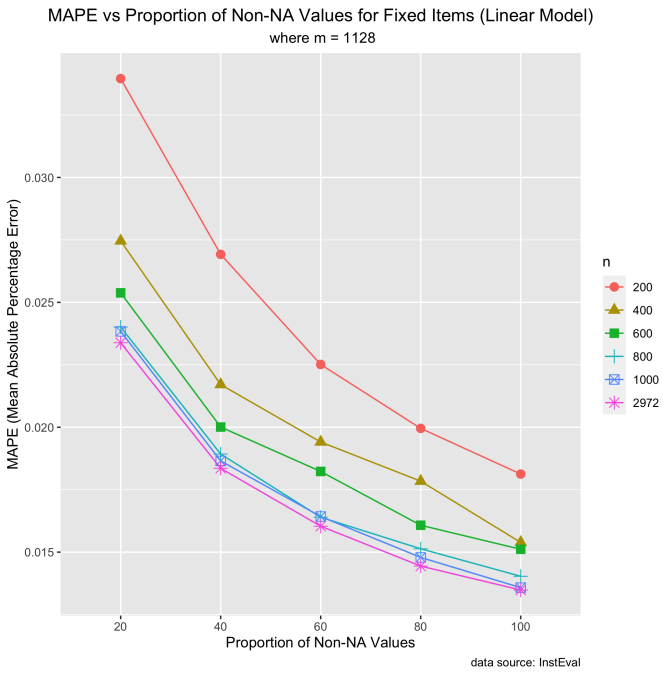

Figure 85: m-1128 (mape)

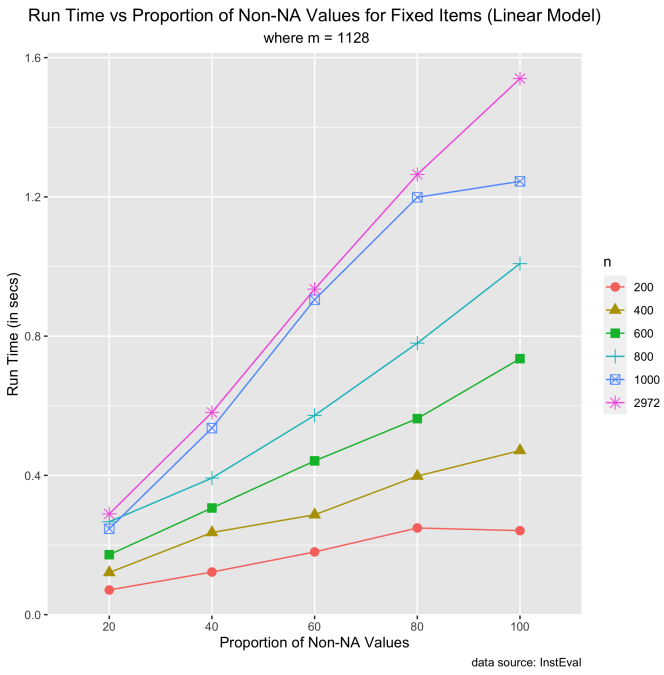

Figure 86: m-1128 (run time)

Fixed N

- Compare across curves: As m increases, curves of MAPE move downward, curves of run time move upward.
- Compare along curves: As N increases, MAPE decreases slightly at the end, but the trend is not as strong as the previous ones. As N increases, run time increases.

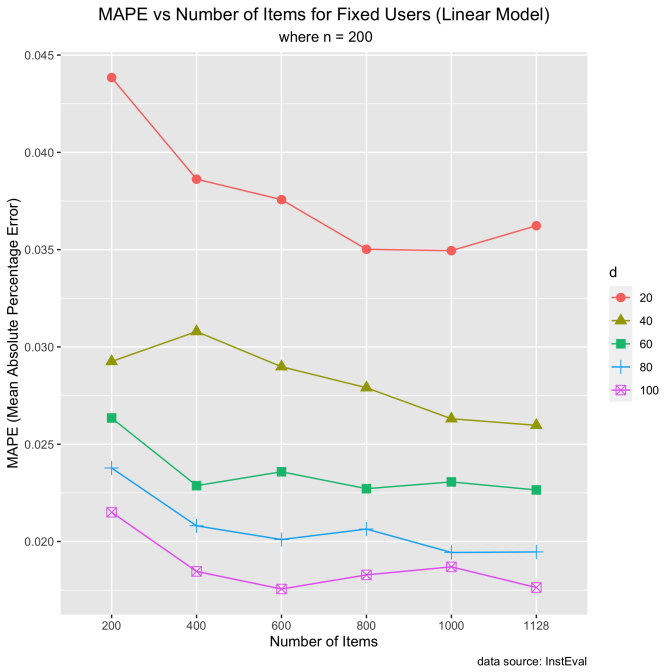

Figure 87: n-200 (mape)

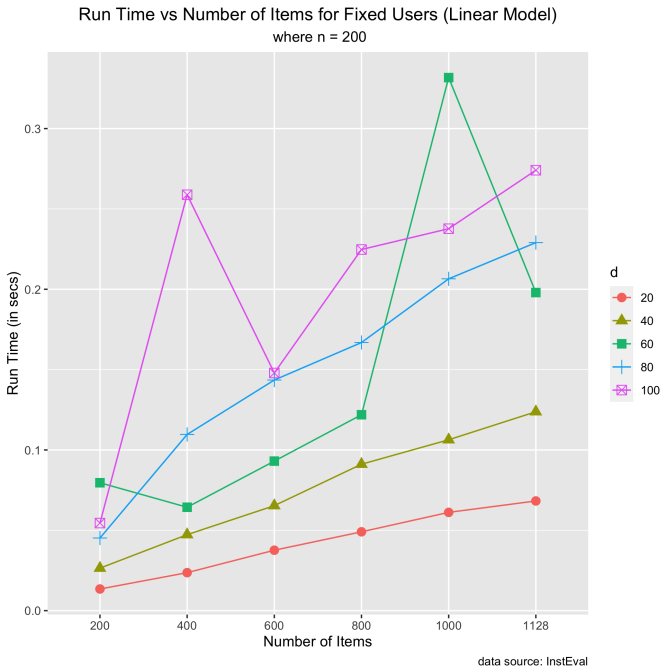

Figure 88: n-200 (run time)

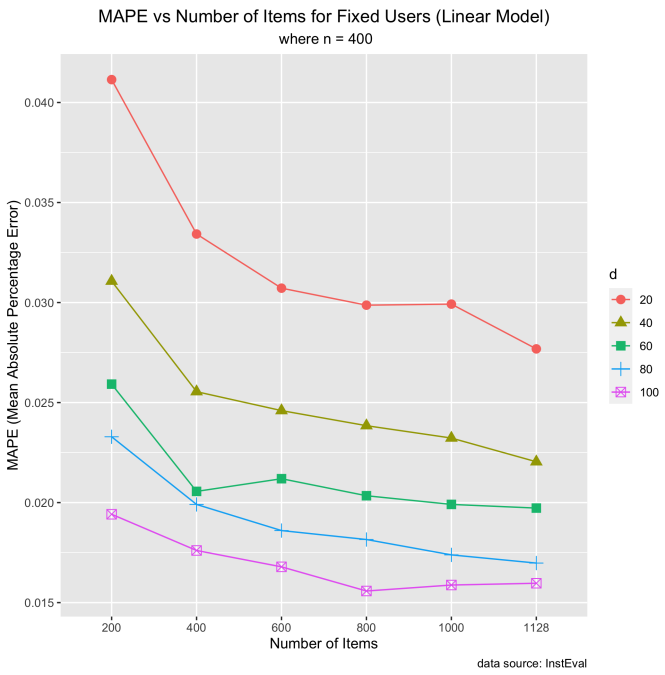

Figure 89: n-400 (mape)

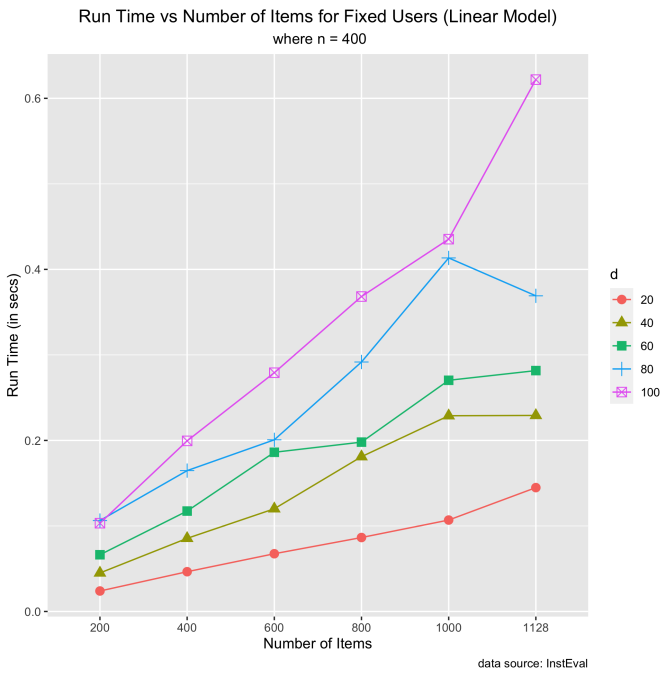

Figure 90: n-400 (run time)

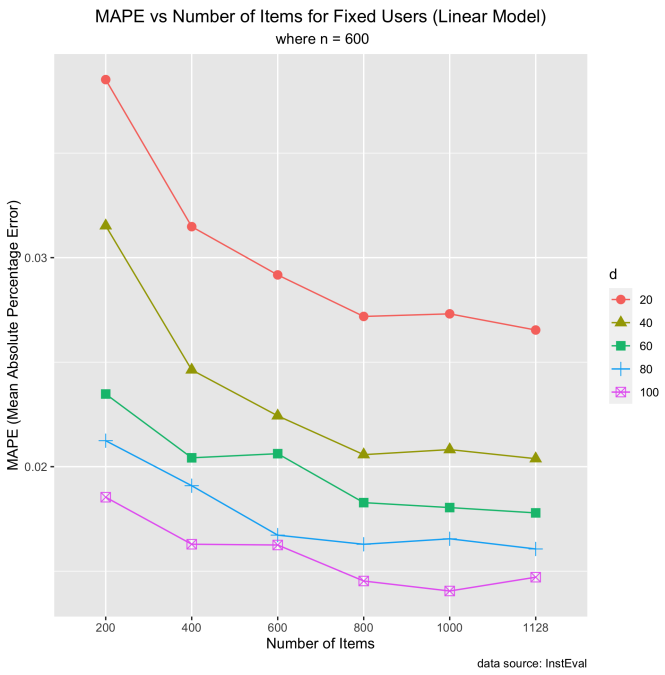

Figure 91: n-600 (mape)

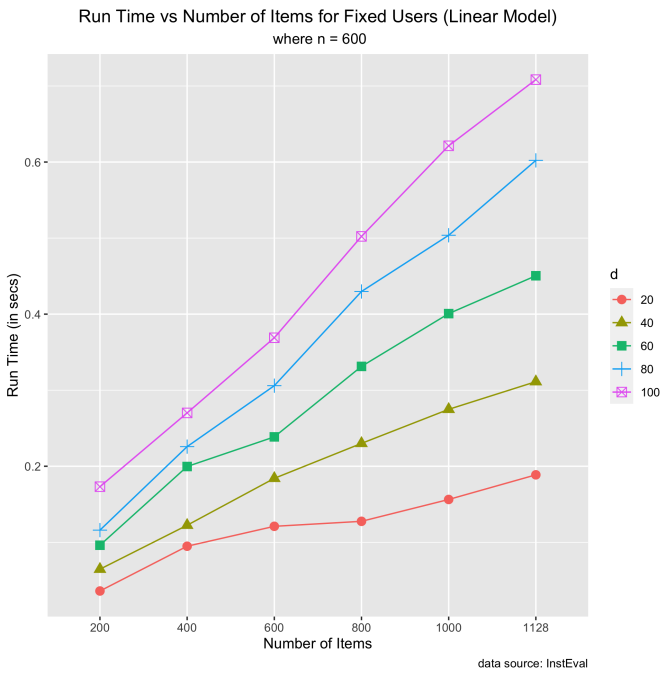

Figure 92: n-600 (run time)

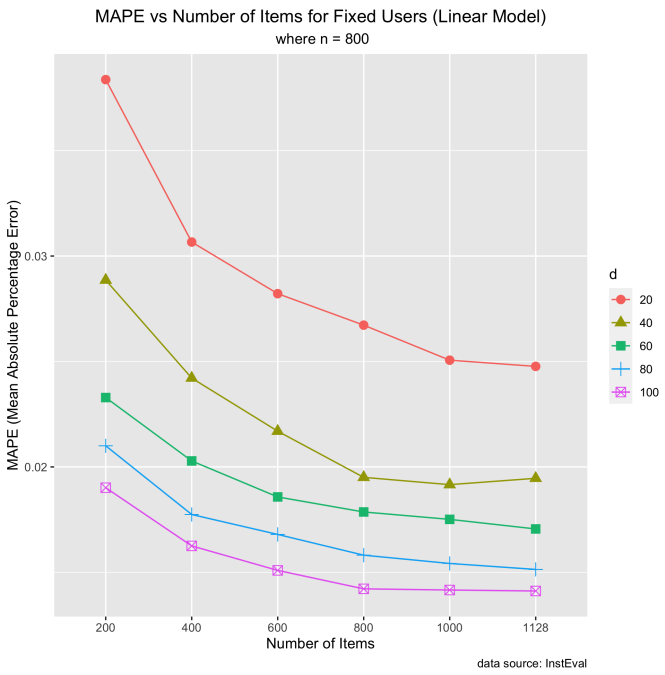

Figure 93: n-800 (mape)

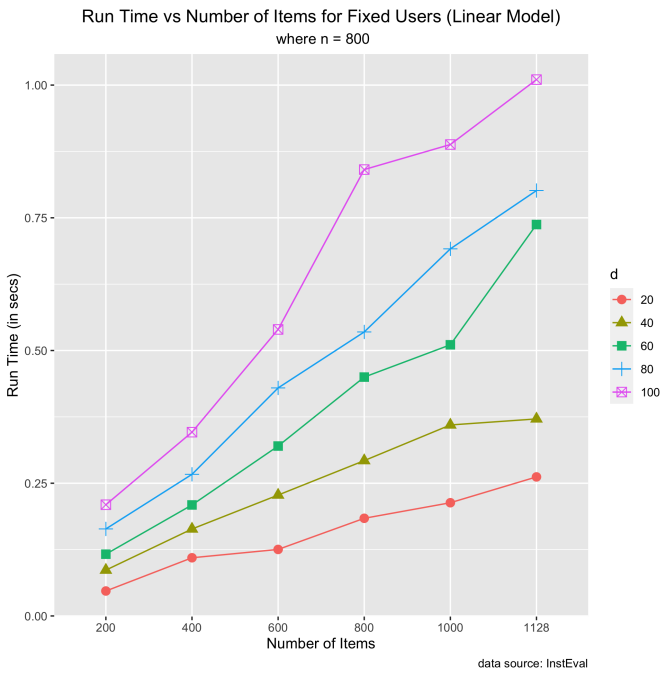

Figure 94: n-800 (run time)

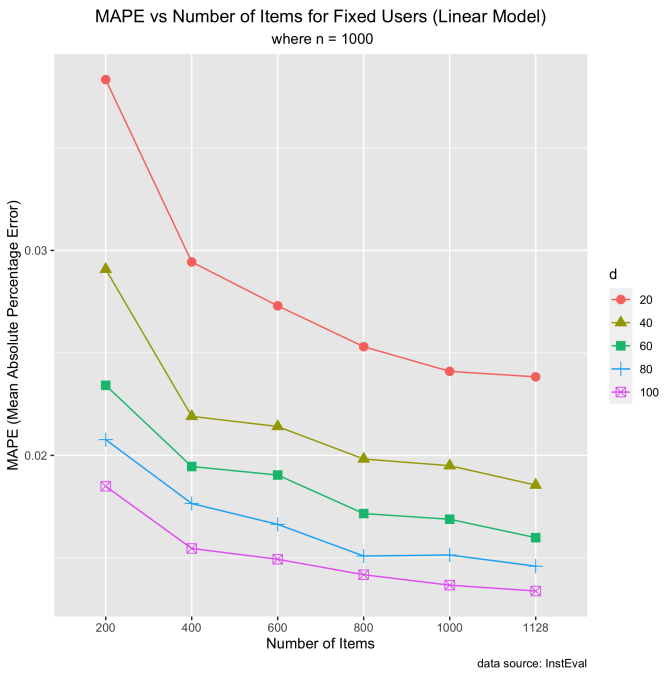

Figure 95: n-1000 (mape)

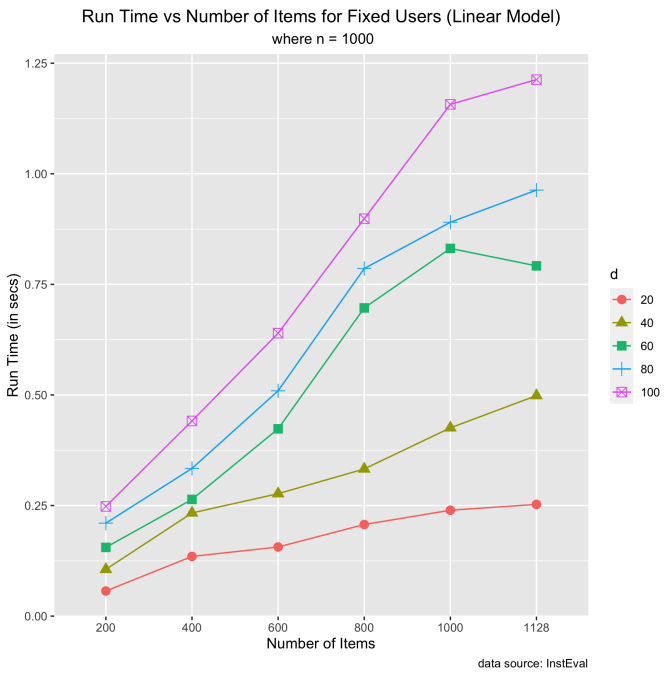

Figure 96: n-1000 (run time)

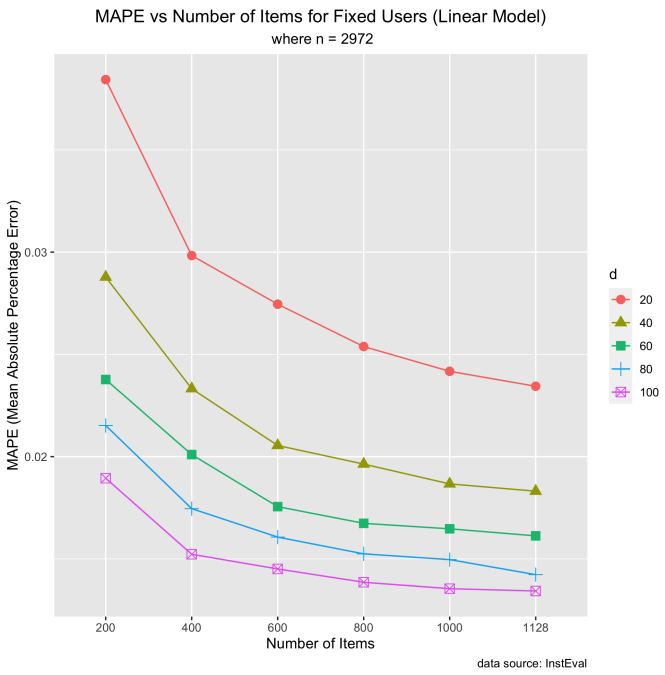

Figure 97: n-1216 (mape)

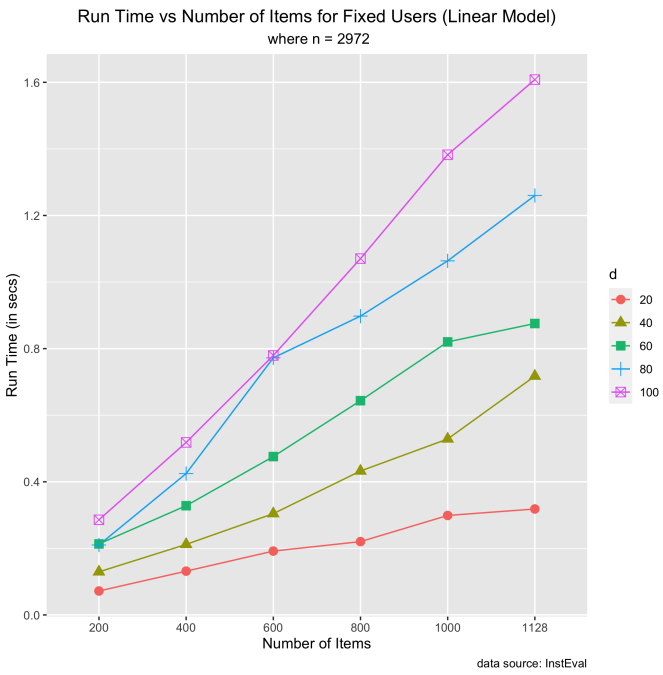

Figure 98: n-1216 (run time)

3.2.2 Matrix Factorization

Fixed D

- Compare across curves: As m increases, curves of MAPE move downward, curves of run time move upward.
- Compare along curves: As N increases, MAPE decreases and run time increases.

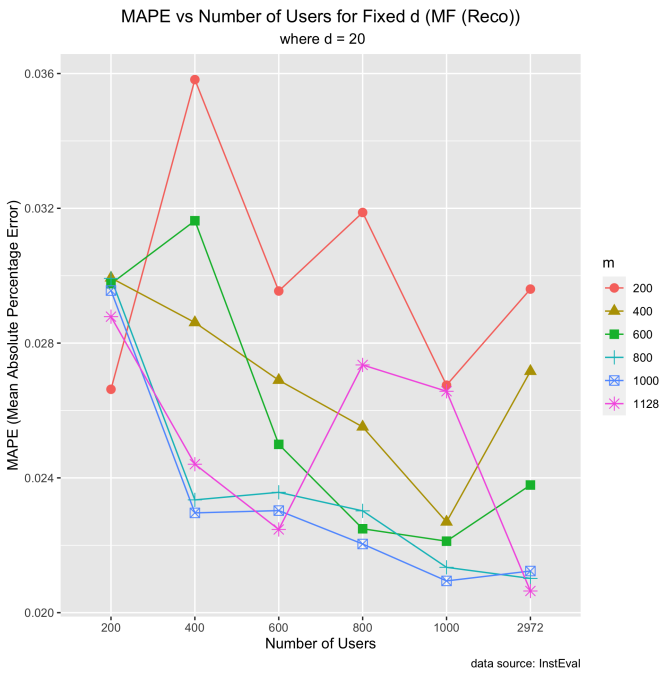

Figure 99: d-20 (mape)

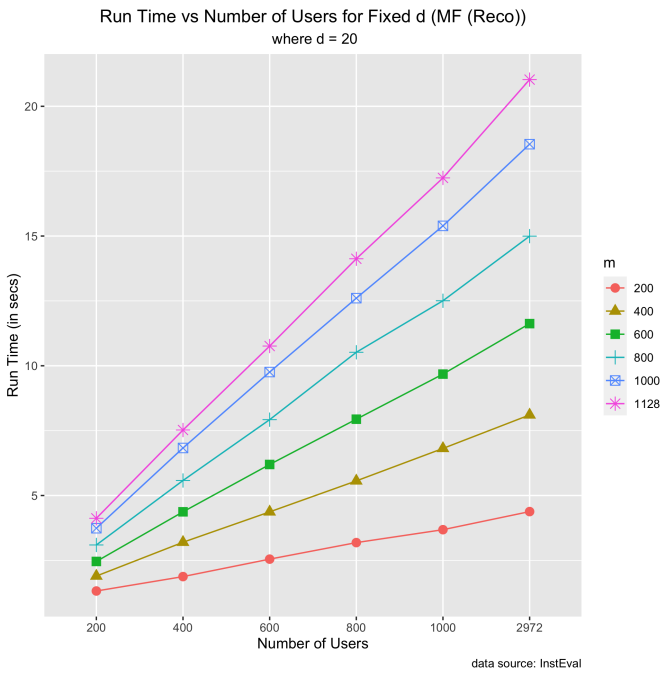

Figure 100: d-20 (run time)

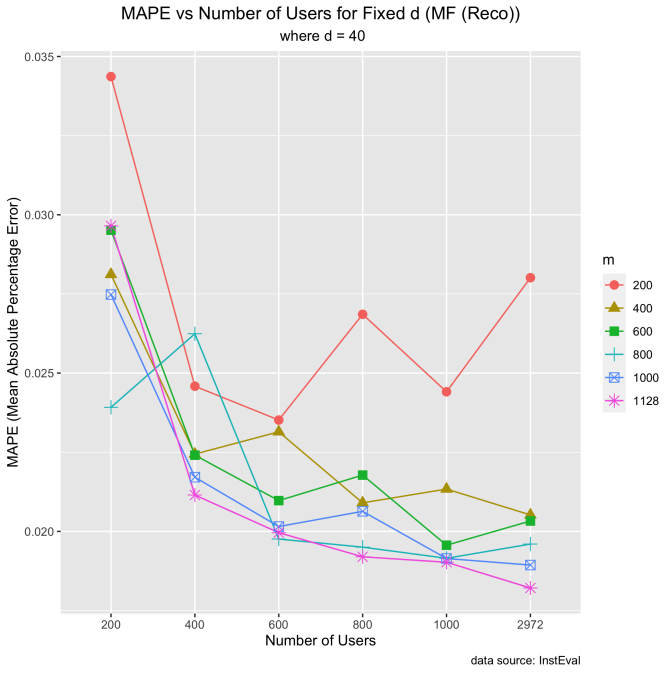

Figure 101: d-40 (mape)

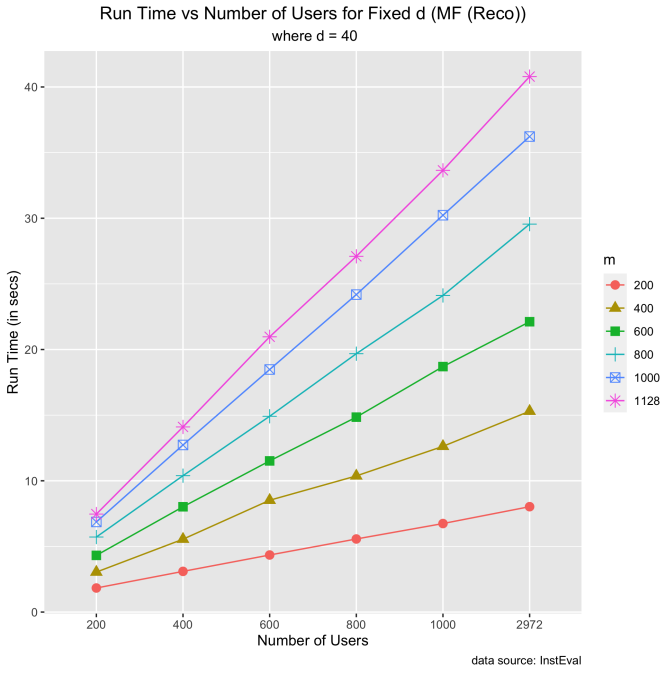

Figure 102: d-40 (run time)

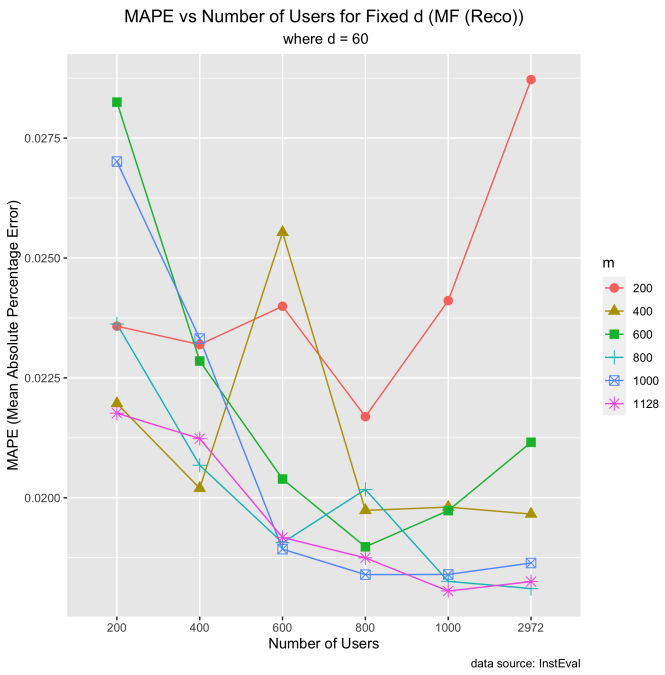

Figure 103: d-60 (mape)

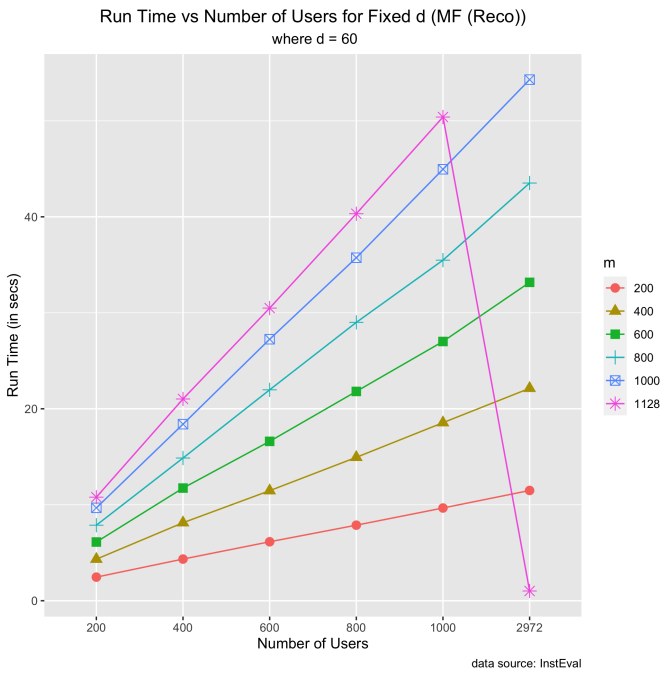

Figure 104: d-60 (run time)

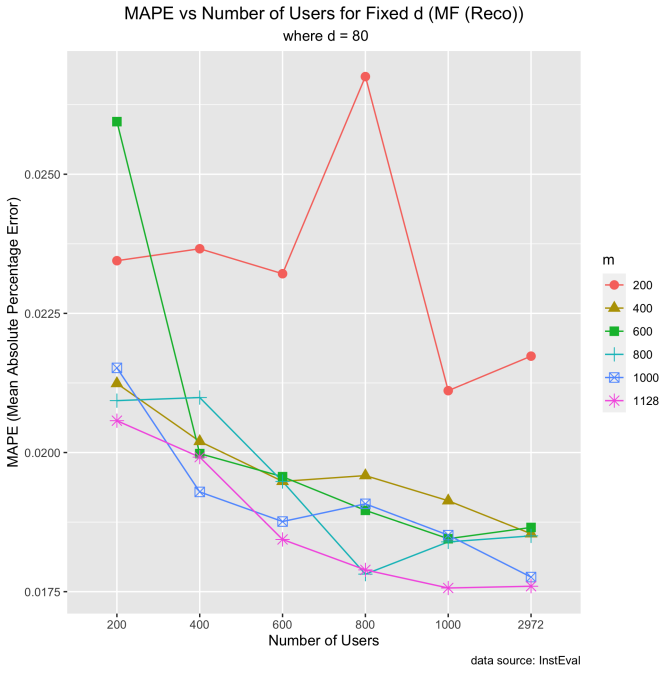

Figure 105: d-80 (mape)

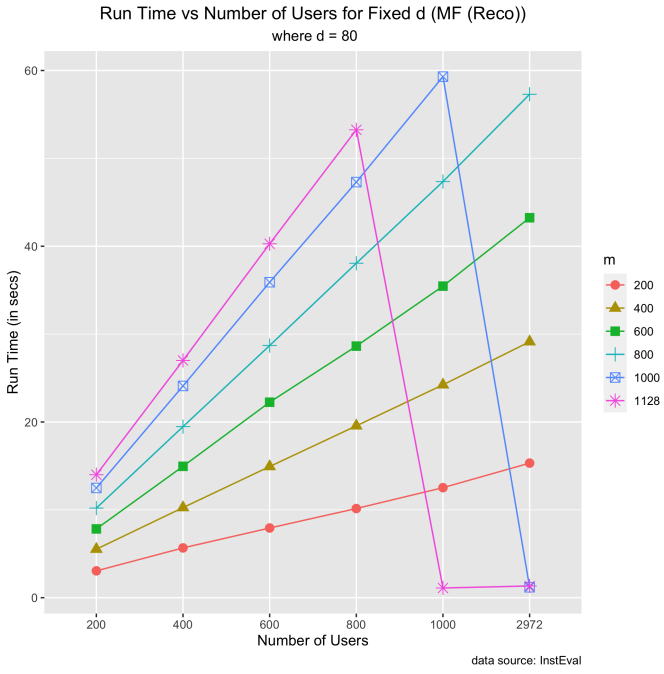

Figure 106: d-80 (run time)

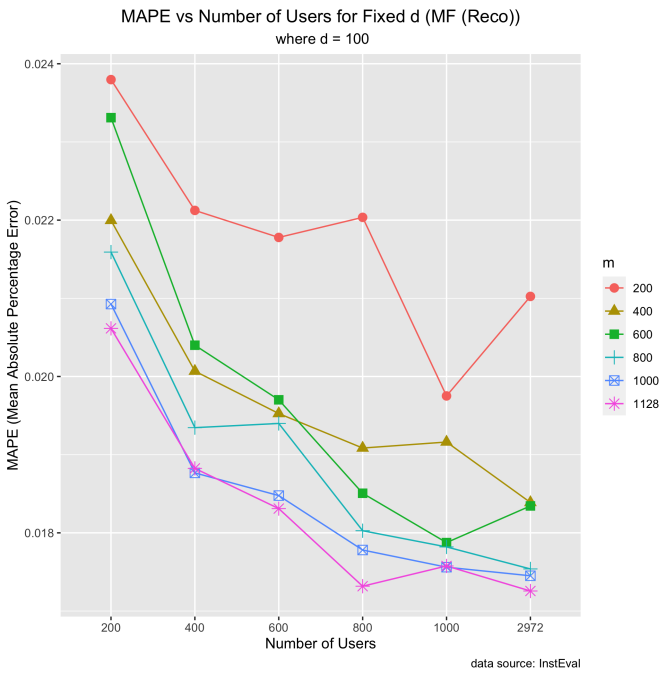

Figure 107: d-100 (mape)

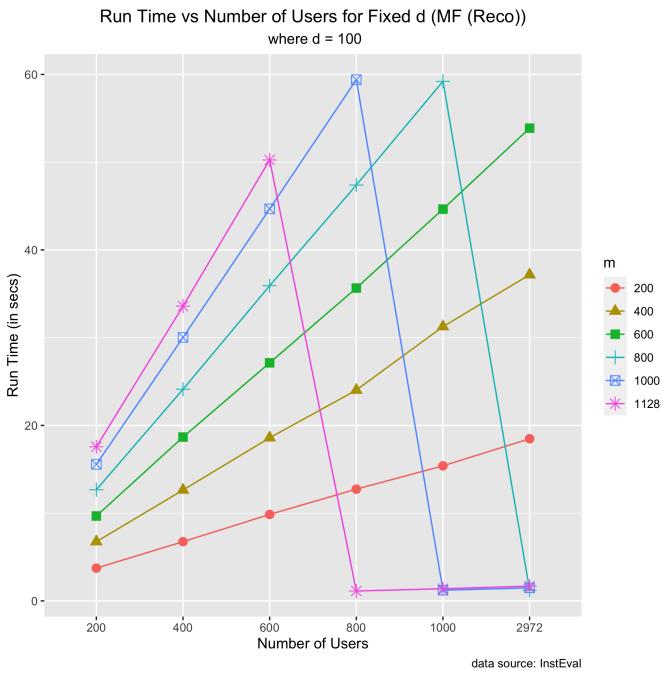

Figure 108: d-100 (run time)

Fixed M

- Compare across curves: As n increases, curves of MAPE move downward, curves of run time move upward.
- Compare along curves: As d increases, MAPE decreases and run time increases.

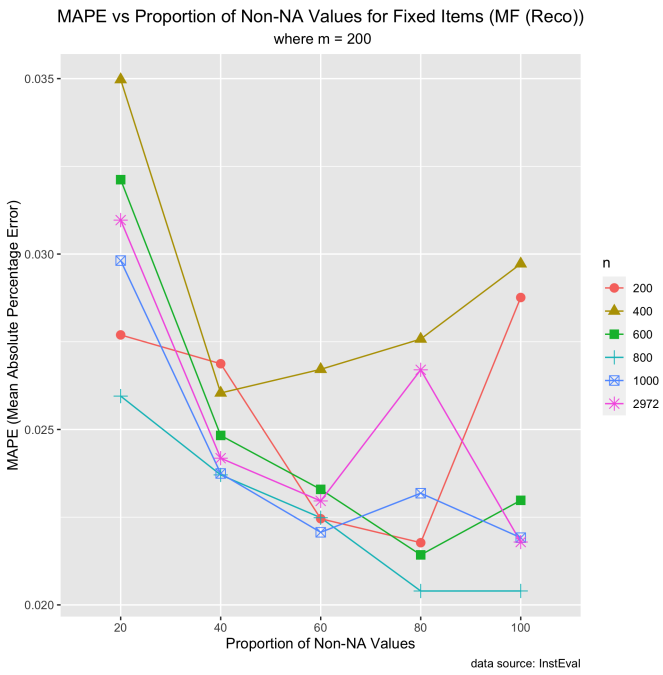

Figure 109: m-200 (mape)

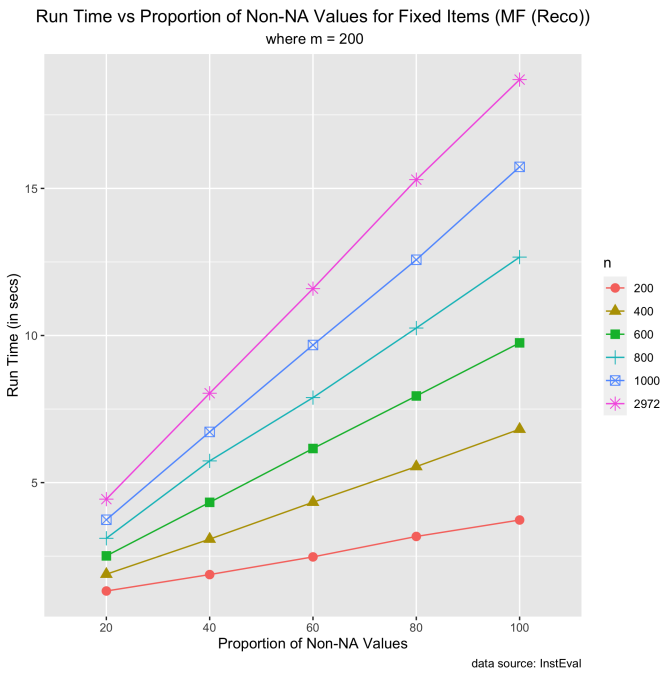

Figure 110: m-200 (run time)

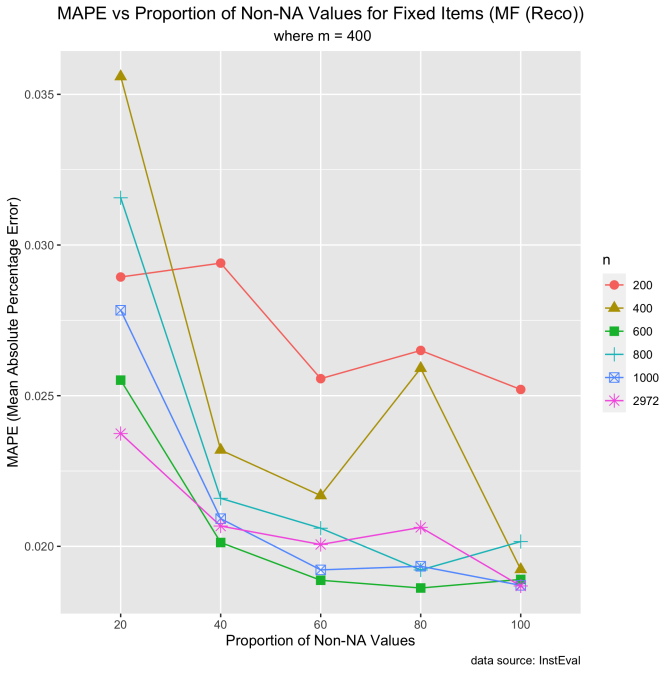

Figure 111: m-400 (mape)

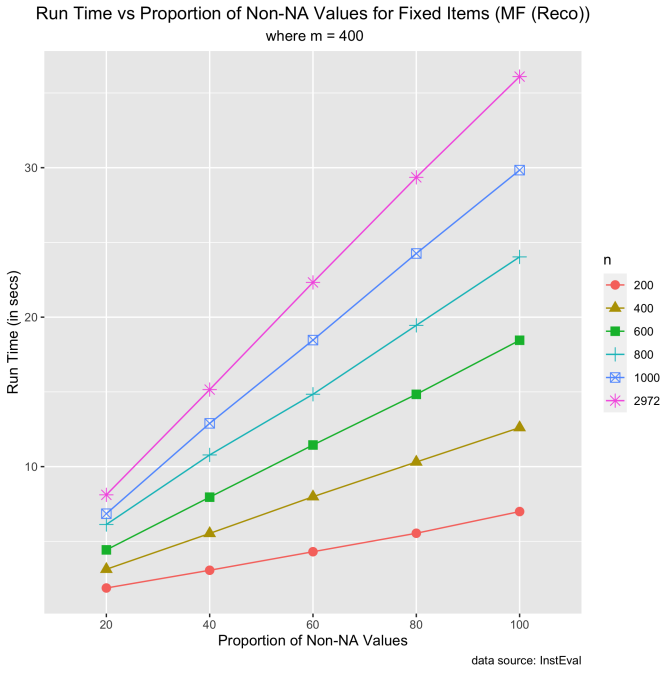

Figure 112: m-400 (run time)

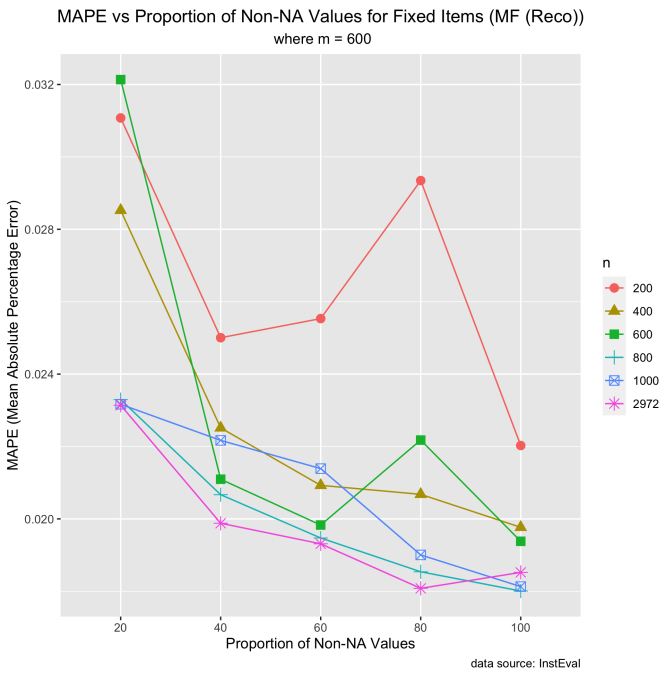

Figure 113: m-600 (mape)

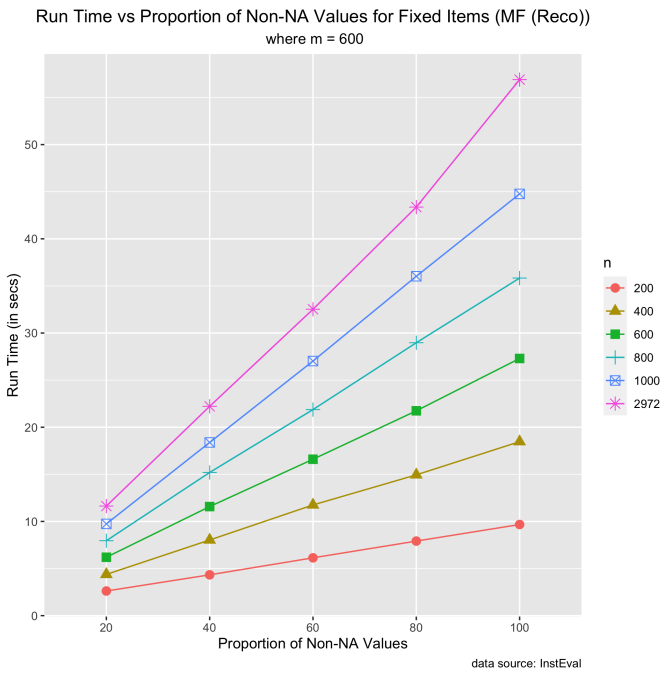

Figure 114: m-600 (run time)

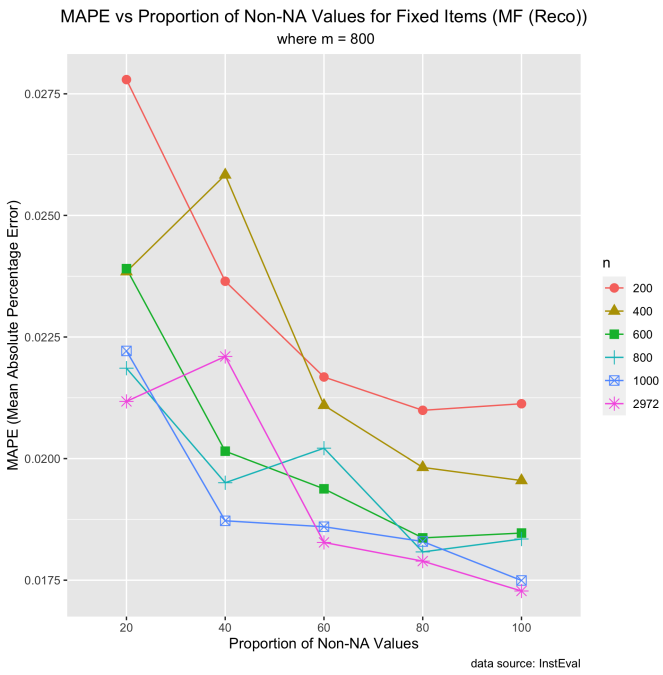

Figure 115: m-800 (mape)

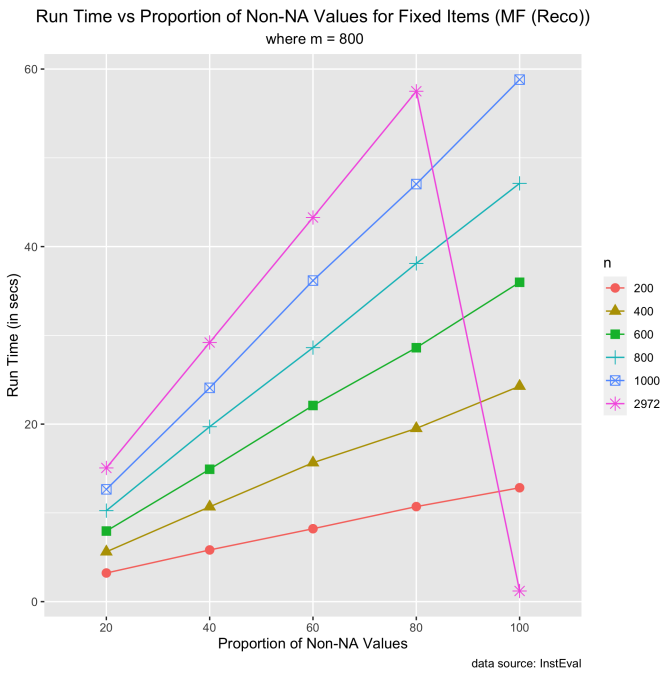

Figure 116: m-800 (run time)

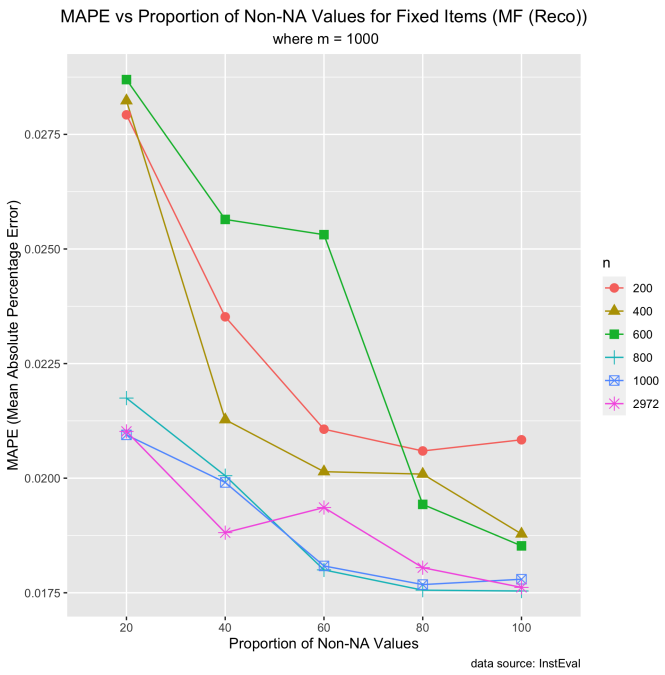

Figure 117: m-1000 (mape)

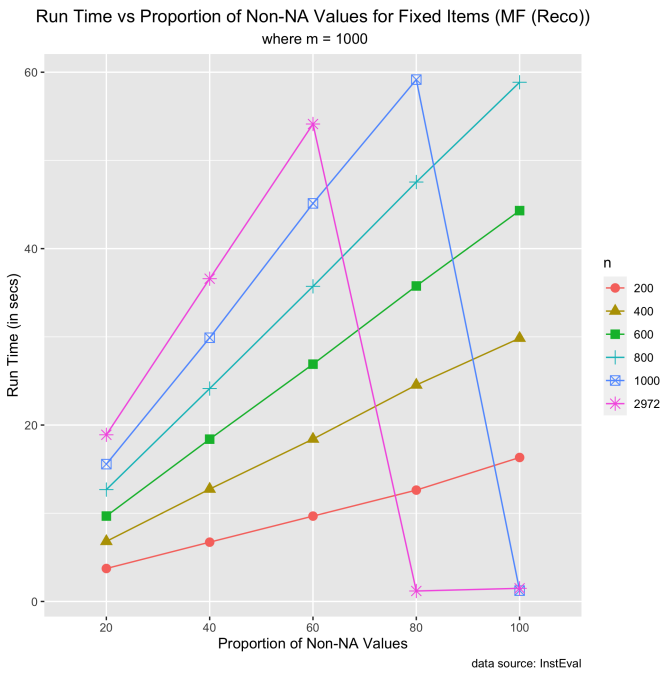

Figure 118: m-1000 (run time)

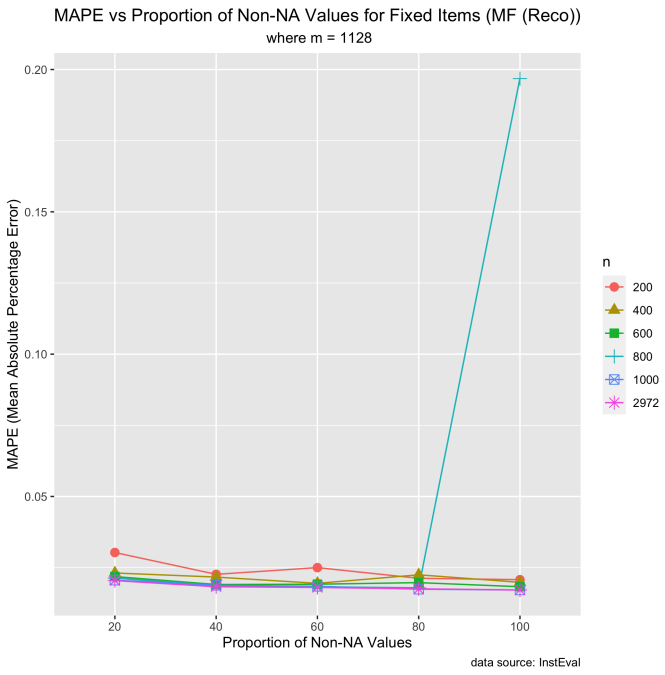

Figure 119: m-1128 (mape)

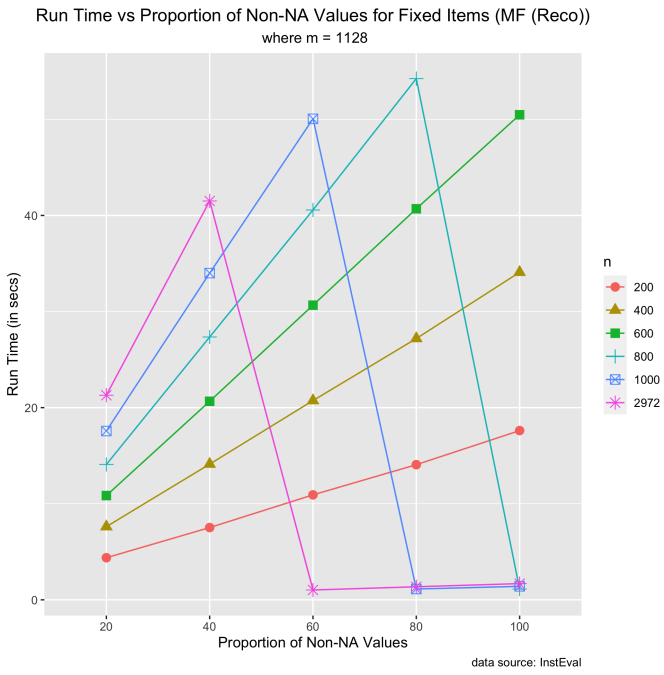

Figure 120: m-1128 (run time)

Fixed N

- Compare across curves: As m increases, curves of MAPE move downward, curves of run time move upward.
- Compare along curves: As N increases, MAPE decreases slightly at the end, but the trend is not as strong as the previous ones. As N increases, run time increases.

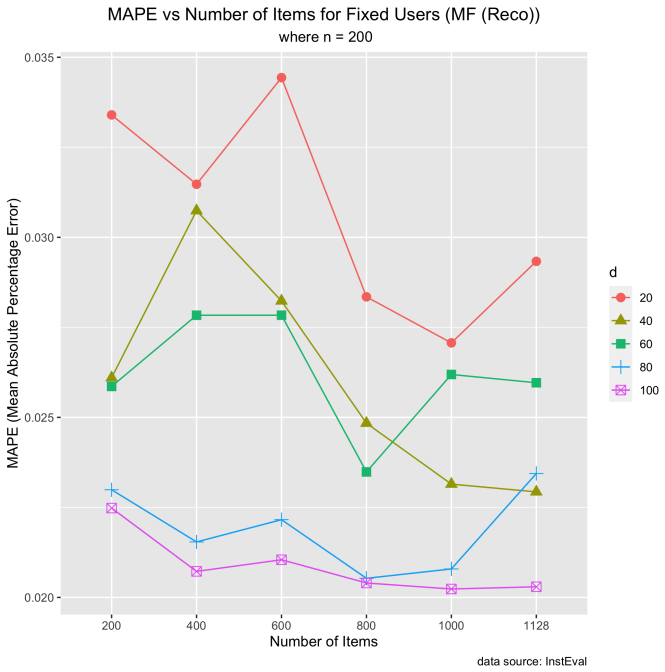

Figure 121: n-200 (mape)

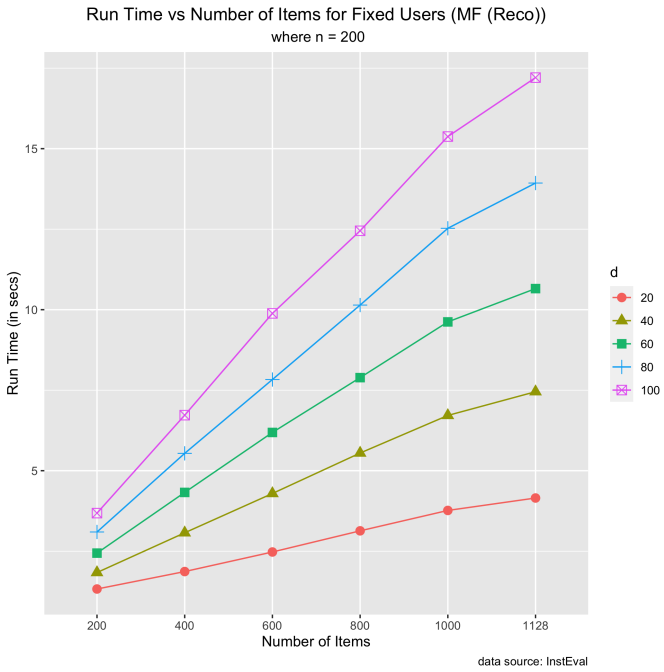

Figure 122: n-200 (run time)

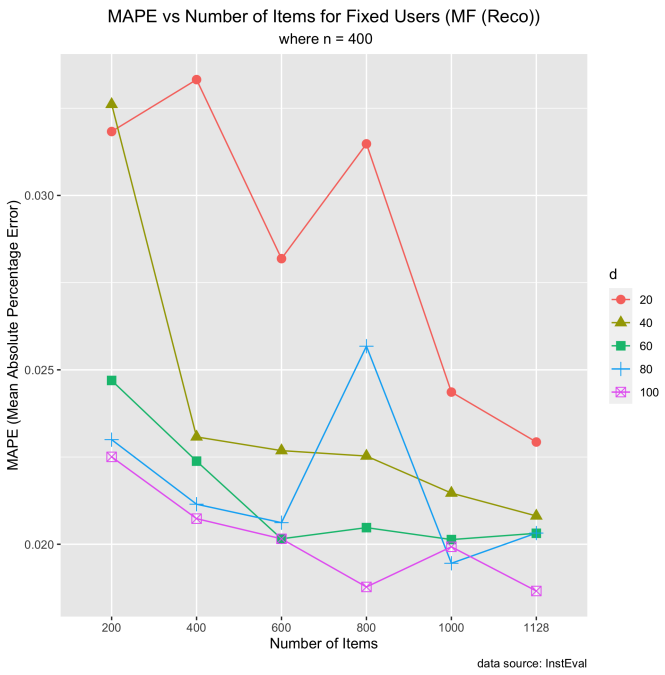

Figure 123: n-400 (mape)

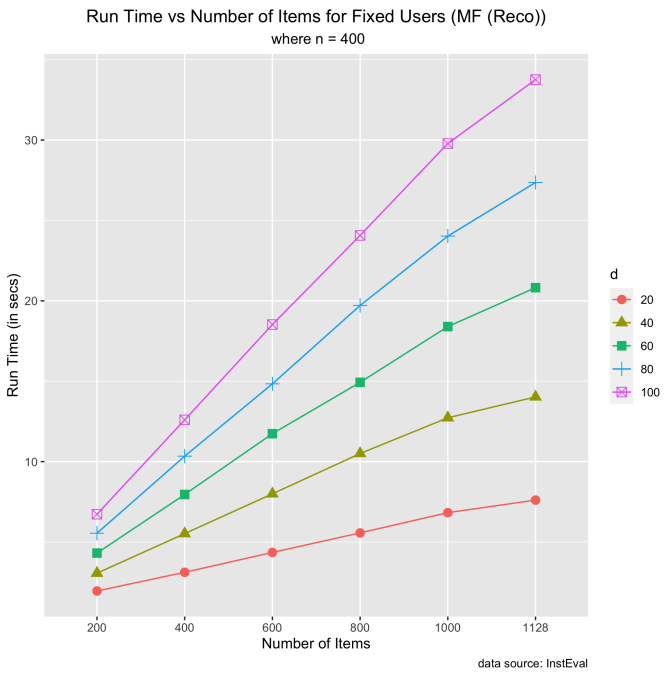

Figure 124: n-400 (run time)

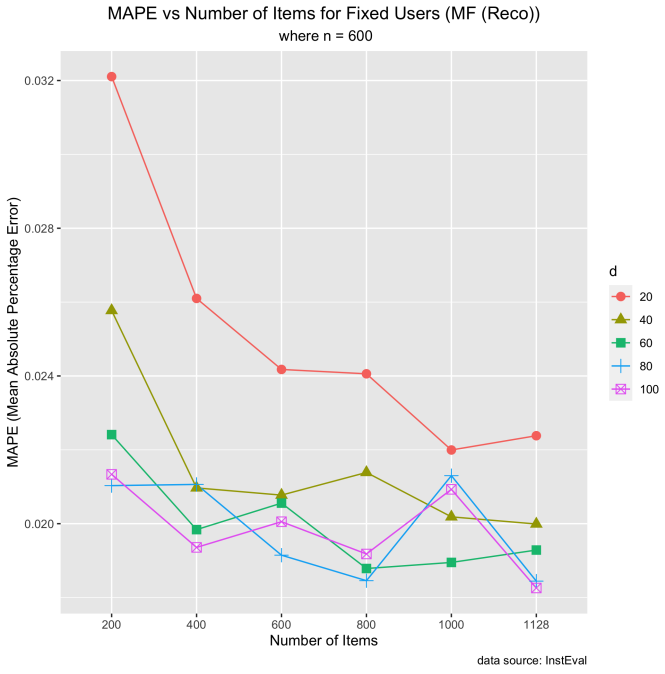

Figure 125: n-600 (mape)

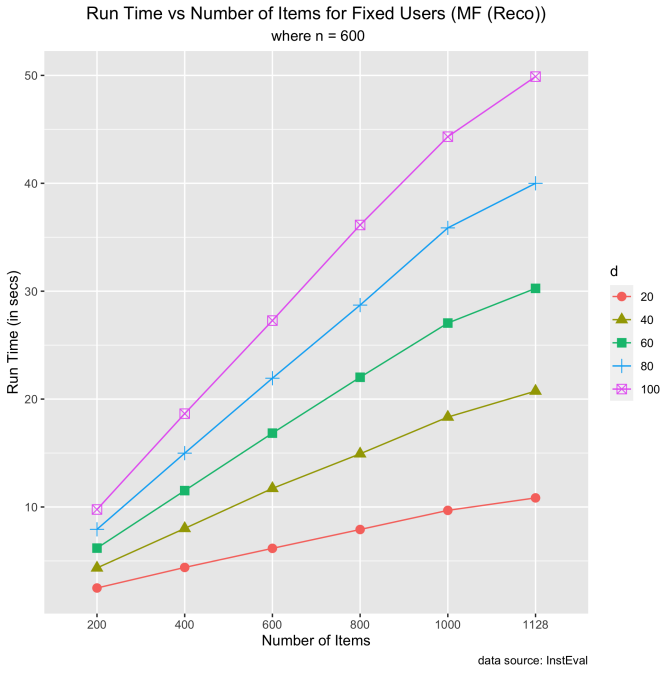

Figure 126: n-600 (run time)

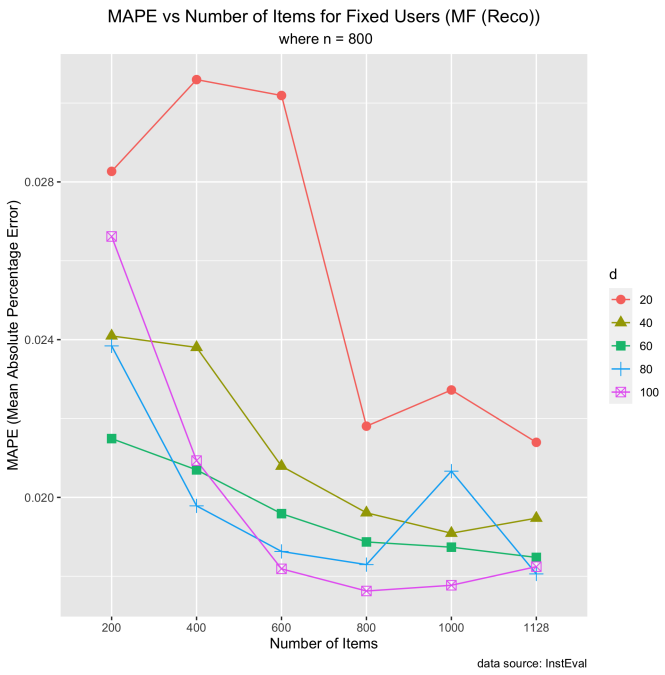

Figure 127: n-800 (mape)

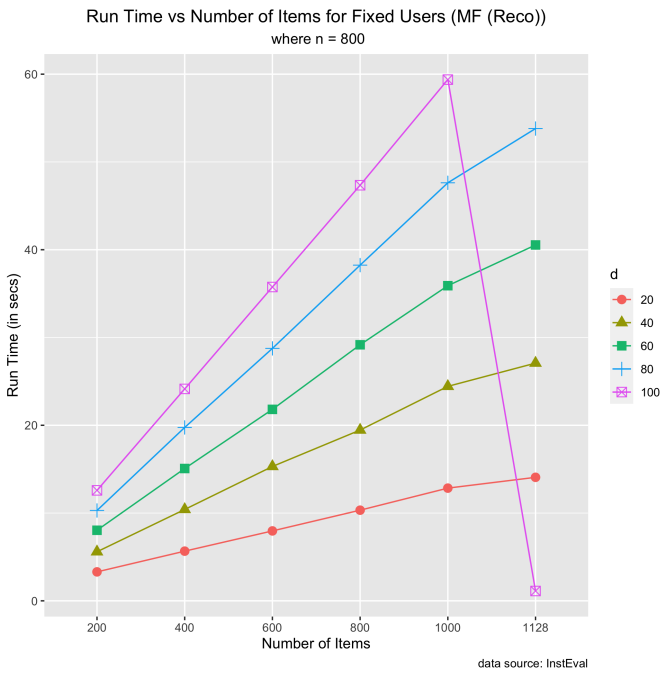

Figure 128: n-800 (run time)

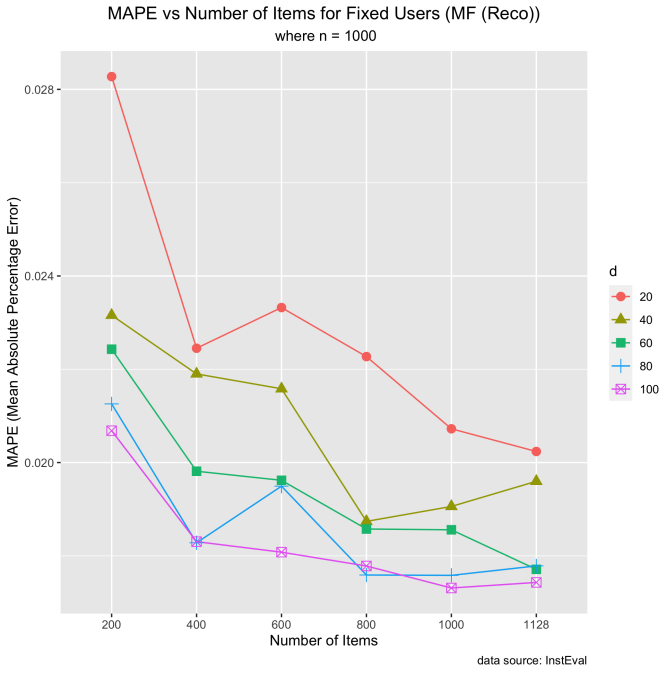

Figure 129: n-1000 (mape)

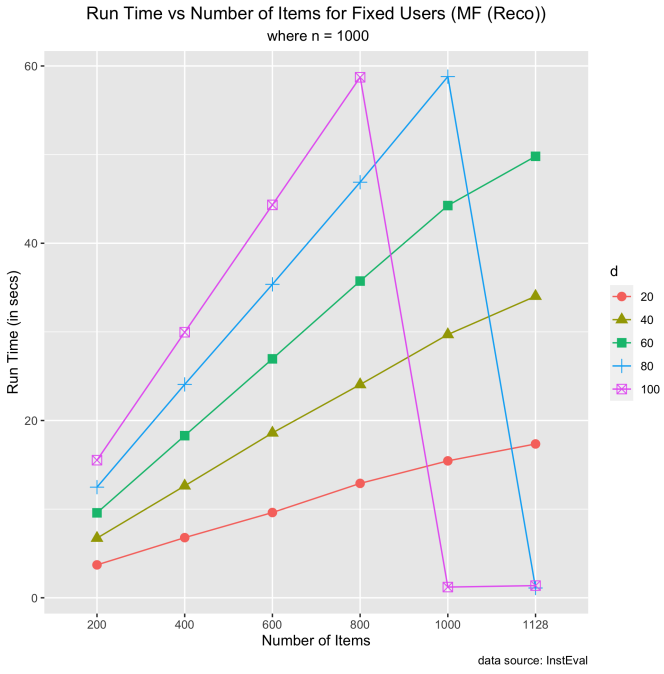

Figure 130: n-1000 (run time)

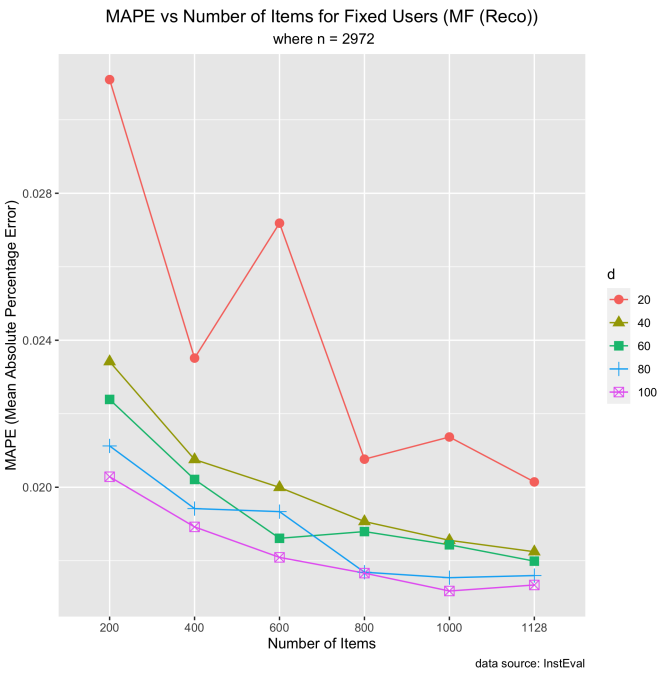

Figure 131: n-1216 (mape)

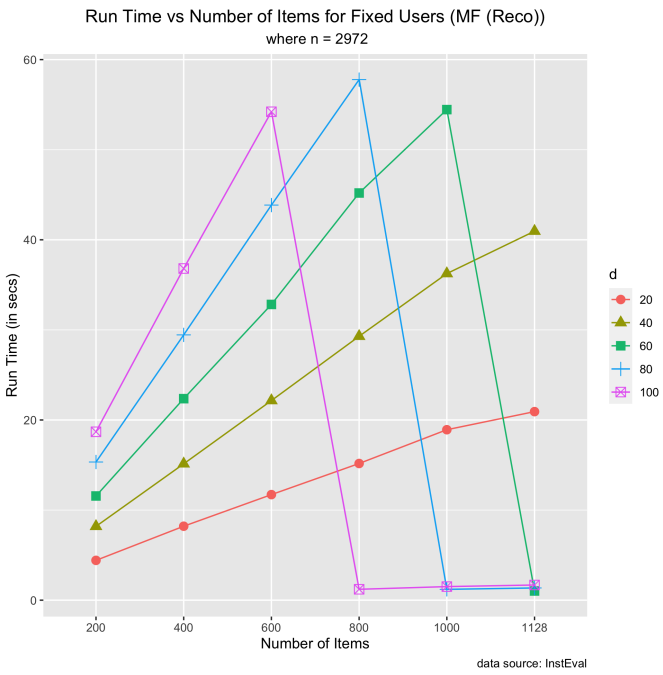

Figure 132: n-1216 (run time)

4 Conclusion

Along the curve: A common trend is that as as N/M/d increases, run time for training the model will increase. And this conclusion is intuitive: more training data requires more training time.

Similarly, as as N/M/d increases, MAPE decreases.

Remark: For InstEval, fixing d: along the curve, increasing N does not improve MAPE as much.

All the stated trend of the curve become unclear in low values of m, n, d.

A Code

```
require(rectools)
require(qeML)
require(ggplot2)
library(recosystem)
load("./data/ml100kpluscovs.RData")
load("./data/InstEval.RData")

# Create The Fake Rating Matrix By Linear Model
generate_fake_rating_matrix <- function(data_frame) {
    user_mean <- tapply(data_frame$userMean, data_frame$user, mean)
    item_mean <- tapply(data_frame$itemMean, data_frame$item, mean)

    model <- qeLin(data_frame[, c("rating", "userMean", "itemMean")],
    "rating")
    user_mean_coef <- matrix(rep(user_mean, each = length(item_mean)),
    ncol = length(item_mean), byrow = TRUE) *
     ↪  model$coefficients["userMean"]
    item_mean_coef <- t(matrix(rep(item_mean, each = length(user_mean)),
    ncol = length(user_mean), byrow = TRUE)) *
     ↪  model$coefficients["itemMean"]

    rating_matrix <- user_mean_coef + item_mean_coef +
    model$coefficients["(Intercept)"]

    colnames(rating_matrix) <- seq_len(length(item_mean))
    rownames(rating_matrix) <- seq_len(length(user_mean))

    for (index in seq_len(length(data_frame$user))) {
        rating_matrix[data_frame$user[index], data_frame$item[index]] <-
        data_frame$rating[index]
    }

    return(list(rating_matrix, user_mean, item_mean))
}

# Generate the Required Subset Rating Matrix
generate_rating_matrix_subset <- function(n, m, d, rating_matrix) {
    orig_row_names <- rownames(rating_matrix)
    orig_col_names <- colnames(rating_matrix)
    rating_matrix_subset <- rating_matrix[1:n, 1:m]

    rating_matrix_subset <- array(rating_matrix_subset)
```

```
    random_sequence <- sample.int(n * m, floor(n * m * (100 - d) / 100))
    rating_matrix_subset[random_sequence] <- NA

    rating_matrix_subset <- matrix(rating_matrix_subset, nrow = n, ncol = m)
    rownames(rating_matrix_subset) <- orig_row_names[1:n]
    colnames(rating_matrix_subset) <- orig_col_names[1:m]

    rating_matrix_subset
}

# Generate Training and Test Sets
generate_training_test_sets <- function(data_frame) {
    test_proportion <- 0.3
    test_index <- sample.int(dim(data_frame)[1],
    floor(dim(data_frame)[1] * test_proportion))

    training_set <- data_frame[-test_index, ]
    test_set <- data_frame[test_index, ]

    return(list(training_set, test_set))
}

# Generate userMean and itemMean Columns
generate_usermean_itemmean <- function(data_frame) {
    user_mean <- tapply(data_frame$ratings, data_frame$userID, mean)
    item_mean <- tapply(data_frame$ratings, data_frame$itemID, mean)

    users <- sort(unique(data_frame$userID))
    items <- sort(unique(data_frame$itemID))

    data_frame$user_mean <- user_mean[match(data_frame$userID, users)]
    data_frame$item_mean <- item_mean[match(data_frame$itemID, items)]

    return(data_frame)
}

# Calculate MAPE Using Linear Model
calculate_mape_linear_model <- function(data_frame_training,
↪   data_frame_test) {
    model <- qeLin(data_frame_training[,
    c("ratings", "user_mean", "item_mean")], "ratings")
    coefficients <- model$coefficients

    user_mean <- tapply(data_frame_training$ratings,
    data_frame_training$userID, mean)
```

```
    item_mean <- tapply(data_frame_training$ratings,
    data_frame_training$itemID, mean)

    users <- sort(unique(data_frame_training$userID))
    items <- sort(unique(data_frame_training$itemID))

    unseen_users <- get_infirst_notsecond(unique(data_frame_test$userID),
    ↪  users)
    unseen_items <- get_infirst_notsecond(unique(data_frame_test$itemID),
    ↪  items)

    users <- c(users, unseen_users)
    items <- c(items, unseen_items)

    user_mean <- c(user_mean, mean(user_mean))
    item_mean <- c(item_mean, mean(item_mean))

    data_frame_test$user_mean <- user_mean[match(data_frame_test$userID,
    ↪  users)]
    data_frame_test$item_mean <- item_mean[match(data_frame_test$itemID,
    ↪  items)]

    num_predictions <- dim(data_frame_test)[1]
    predictions <- rep(coefficients["(Intercept)"], each = num_predictions)
    ↪  +
    rep(coefficients["user_mean"], each = num_predictions) *
    data_frame_test$user_mean +
    rep(coefficients["item_mean"], each = num_predictions) *
    data_frame_test$item_mean

    actual_ratings <- data_frame_test$ratings
    mape <- mean(abs((actual_ratings - predictions) / actual_ratings))

    return(mape)
}

# Get Elements In The First Vector But Not The Second Vector
get_infirst_notsecond <- function(first_vector, second_vector) {
    unseen_element <- c()

    for (element in first_vector) {
        if (!(element %in% second_vector)) {
            unseen_element <- c(unseen_element, element)
        }
    }
```

```
    return(unseen_element)
}

# Generate Training and Test Files For Matrix Factorization
generate_training_test_files <- function(training_test_sets) {
    write.table(training_test_sets[[1]], "train.txt",
    row.names = FALSE, col.names = FALSE)
    write.table(training_test_sets[[2]][, -3], "test.txt",
    row.names = FALSE, col.names = FALSE)

    return(list("train.txt", "test.txt"))
}

# Calculate MAPE Using Matrix Factorization (Recosystem)
calculate_mape_mf_reco <- function(train_file, test_file, actual_ratings) {
    train_set <- data_file(train_file)
    test_set <- data_file(test_file)

    r <- Reco()
    opts <- r$tune(train_set, opts = list(dim = c(10, 20, 30),
    lrate = c(0.1, 0.2), costp_l1 = 0, costq_l1 = 0, nthread = 1, niter =
    ↪  10))
    r$train(train_set, opts = c(opts$min, nthread = 1, niter = 10))

    pred_file <- tempfile()
    r$predict(test_set, out_file(pred_file))

    file.remove(train_file)
    file.remove(test_file)

    return(mean(abs((scan(pred_file) - actual_ratings) / actual_ratings)))
}

# Generate Graph For Fixed d
generate_graph_for_fixed_d <- function(d, mape_run_time_m_n,
model, data_source) {
    mape_m_n <- mape_run_time_m_n[[1]]
    run_time_m_n <- mape_run_time_m_n[[2]]

    data_frame_m_n <- data.frame(matrix(nrow = length(n) * length(m),
    ncol = 4))
    colnames(data_frame_m_n) <- c("n", "m", "mape", "run_time")
    data_frame_m_n$n <- as.factor(array(matrix(rep(n, each = length(m)),
    ncol = length(m), byrow = TRUE)))
```

```
data_frame_m_n$m <- as.factor(rep(m, each = length(n)))
mape <- c()
for (i in seq_len(length(m))) {
    mape <- c(mape, mape_m_n[i, ])
}
data_frame_m_n$mape <- mape
run_time <- c()
for (i in seq_len(length(m))) {
    run_time <- c(run_time, run_time_m_n[i, ])
}
data_frame_m_n$run_time <- run_time

plot <- ggplot(data = data_frame_m_n, aes(x = n, y = mape, group = m,
color = m, fill = m, shape = m)) +
    geom_line() +
    geom_point(size = 3) +
    labs(
        title = paste0("MAPE vs Number of Users for Fixed d (", model,
        ↪  ")"),
        subtitle = paste0(
                " where d = ",
                d
            ),
        x = "Number of Users",
        y = "MAPE (Mean Absolute Percentage Error)",
        caption = paste0("data source: ", data_source)
    ) +
    theme(plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5))

save_path <- paste0("./results/", data_source, "/", model)
ggsave(paste0(save_path, "/d-", d, " (mape)", ".png"), plot)

plot <- ggplot(data = data_frame_m_n, aes(x = n, y = run_time, group =
↪  m,
color = m, fill = m, shape = m)) +
    geom_line() +
    geom_point(size = 3) +
    labs(
        title = paste0("Run Time vs Number of Users for Fixed d (",
        ↪  model, ")"),
        subtitle = paste0(
                " where d = ",
                d
            ),
```

```
            x = "Number of Users",
            y = "Run Time (in secs)",
            caption = paste0("data source: ", data_source)
        ) +
        theme(plot.title = element_text(hjust = 0.5),
              plot.subtitle = element_text(hjust = 0.5))

    save_path <- paste0("./results/", data_source, "/", model)
    ggsave(paste0(save_path, "/d-", d, " (run time)", ".png"), plot)
}

# Generate Graph For Fixed n
generate_graph_for_fixed_n <- function(n, mape_run_time_d_m,
model, data_source) {
    mape_d_m <- mape_run_time_d_m[[1]]
    run_time_d_m <- mape_run_time_d_m[[2]]

    data_frame_d_m <- data.frame(matrix(nrow = length(m) * length(d),
    ncol = 4))
    colnames(data_frame_d_m) <- c("m", "d", "mape", "run_time")
    data_frame_d_m$m <- as.factor(array(matrix(rep(m, each = length(d)),
    ncol = length(d), byrow = TRUE)))
    data_frame_d_m$d <- as.factor(rep(d, each = length(m)))
    mape <- c()
    for (i in seq_len(length(d))) {
        mape <- c(mape, mape_d_m[i, ])
    }
    data_frame_d_m$mape <- mape
    run_time <- c()
    for (i in seq_len(length(d))) {
        run_time <- c(run_time, run_time_d_m[i, ])
    }
    data_frame_d_m$run_time <- run_time

    plot <- ggplot(data = data_frame_d_m, aes(x = m, y = mape, group = d,
    color = d, fill = d, shape = d)) +
        geom_line() +
        geom_point(size = 3) +
        labs(
            title = paste0("MAPE vs Number of Items for Fixed Users (",
            ↪  model, ")"),
            subtitle = paste0(
                    " where n = ",
                    n
                ),
```

```
            x = "Number of Items",
            y = "MAPE (Mean Absolute Percentage Error)",
            caption = paste0("data source: ", data_source)
        ) +
        theme(plot.title = element_text(hjust = 0.5),
              plot.subtitle = element_text(hjust = 0.5))

    save_path <- paste0("./results/", data_source, "/", model)
    ggsave(paste0(save_path, "/n-", n, " (mape)", ".png"), plot)

    plot <- ggplot(data = data_frame_d_m, aes(x = m, y = run_time, group =
    ↪  d,
    color = d, fill = d, shape = d)) +
        geom_line() +
        geom_point(size = 3) +
        labs(
            title = paste0("Run Time vs Number of Items for Fixed Users (",
            ↪  model, ")"),
            subtitle = paste0(
                    " where n = ",
                    n
                ),
            x = "Number of Items",
            y = "Run Time (in secs)",
            caption = paste0("data source: ", data_source)
        ) +
        theme(plot.title = element_text(hjust = 0.5),
              plot.subtitle = element_text(hjust = 0.5))

    save_path <- paste0("./results/", data_source, "/", model)
    ggsave(paste0(save_path, "/n-", n, " (run time)", ".png"), plot)
}

# Generate Graph For Fixed m
generate_graph_for_fixed_m <- function(m, mape_run_time_n_d,
model, data_source) {
    mape_n_d <- mape_run_time_n_d[[1]]
    run_time_n_d <- mape_run_time_n_d[[2]]

    data_frame_n_d <- data.frame(matrix(nrow = length(d) * length(n),
    ncol = 4))
    colnames(data_frame_n_d) <- c("d", "n", "mape", "run_time")
    data_frame_n_d$d <- as.factor(array(matrix(rep(d, each = length(n)),
    ncol = length(n), byrow = TRUE)))
    data_frame_n_d$n <- as.factor(rep(n, each = length(d)))
```

```
mape <- c()
for (i in seq_len(length(n))) {
    mape <- c(mape, mape_n_d[i, ])
}
data_frame_n_d$mape <- mape
run_time <- c()
for (i in seq_len(length(n))) {
    run_time <- c(run_time, run_time_n_d[i, ])
}
data_frame_n_d$run_time <- run_time

plot <- ggplot(data = data_frame_n_d, aes(x = d, y = mape, group = n,
color = n, fill = n, shape = n)) +
    geom_line() +
    geom_point(size = 3) +
    labs(
        title = paste0("MAPE vs Proportion of Non-NA Values for Fixed
        ↪  Items (", model, ")"),
        subtitle = paste0(
                " where m = ",
                m
            ),
        x = "Proportion of Non-NA Values",
        y = "MAPE (Mean Absolute Percentage Error)",
        caption = paste0("data source: ", data_source)
    ) +
    theme(plot.title = element_text(hjust = 0.5),
          plot.subtitle = element_text(hjust = 0.5))

save_path <- paste0("./results/", data_source, "/", model)
ggsave(paste0(save_path, "/m-", m, " (mape)", ".png"), plot)

plot <- ggplot(data = data_frame_n_d, aes(x = d, y = run_time, group =
↪  n,
color = n, fill = n, shape = n)) +
    geom_line() +
    geom_point(size = 3) +
    labs(
        title = paste0("Run Time vs Proportion of Non-NA Values for
        ↪  Fixed Items (", model, ")"),
        subtitle = paste0(
                " where m = ",
                m
            ),
        x = "Proportion of Non-NA Values",
```

```
            y = "Run Time (in secs)",
            caption = paste0("data source: ", data_source)
        ) +
        theme(plot.title = element_text(hjust = 0.5),
              plot.subtitle = element_text(hjust = 0.5))

    save_path <- paste0("./results/", data_source, "/", model)
    ggsave(paste0(save_path, "/m-", m, " (run time)", ".png"), plot)
}

# Calculate MAPE For Fixed d
calculate_mape_fixed_d <- function(d, model) {
    mape_m_n <- matrix(, nrow = length(m), ncol = length(n))
    run_time_m_n <- matrix(, nrow = length(m), ncol = length(n))

    for (m_choice in m) {
        mape_n <- c()
        run_time_n <- c()

        for (n_choice in n) {
            rating_matrix_subset <- generate_rating_matrix_subset(
                n_choice, m_choice, d, rating_matrix
            )

            data_frame <- toUserItemRatings(rating_matrix_subset)
            data_frame$userID <- as.numeric(data_frame$userID)
            data_frame$itemID <- as.numeric(data_frame$itemID)
            training_test_sets <- generate_training_test_sets(data_frame)

            if (model == "Linear Model") {
                training_set <- generate_usermean_itemmean(
                    training_test_sets[[1]])
                start_time <- Sys.time()
                mape <- calculate_mape_linear_model(training_set,
                training_test_sets[[2]])
                end_time <- Sys.time()
                run_time <- end_time - start_time
            }

            if (model == "MF (Reco)") {
                training_test_files <- generate_training_test_files(
                training_test_sets)
                start_time <- Sys.time()
                mape <- calculate_mape_mf_reco(training_test_files[[1]],
                training_test_files[[2]], training_test_sets[[2]]$ratings)
```

```
            end_time <- Sys.time()
            run_time <- end_time - start_time
        }

        mape_n <- c(mape_n, mape)
        run_time_n <- c(run_time_n, run_time)
    }
    mape_m_n[match(m_choice, m), ] <- mape_n
    run_time_m_n[match(m_choice, m), ] <- run_time_n
}

return(list(mape_m_n, run_time_m_n))
}

# Calculate MAPE For Fixed n
calculate_mape_fixed_n <- function(n, model) {
    mape_d_m <- matrix(, nrow = length(d), ncol = length(m))
    run_time_d_m <- matrix(, nrow = length(d), ncol = length(m))

    for (d_choice in d) {
        mape_m <- c()
        run_time_m <- c()

        for (m_choice in m) {
            rating_matrix_subset <- generate_rating_matrix_subset(
                n, m_choice, d_choice, rating_matrix
            )

            data_frame <- toUserItemRatings(rating_matrix_subset)
            data_frame$userID <- as.numeric(data_frame$userID)
            data_frame$itemID <- as.numeric(data_frame$itemID)
            training_test_sets <- generate_training_test_sets(data_frame)

            if (model == "Linear Model") {
                training_set <- generate_usermean_itemmean(
                    training_test_sets[[1]])
                start_time <- Sys.time()
                mape <- calculate_mape_linear_model(training_set,
                training_test_sets[[2]])
                end_time <- Sys.time()
                run_time <- end_time - start_time
            }

            if (model == "MF (Reco)") {
                training_test_files <- generate_training_test_files(
```

```
                training_test_sets)
                start_time <- Sys.time()
                mape <- calculate_mape_mf_reco(training_test_files[[1]],
                training_test_files[[2]], training_test_sets[[2]]$ratings)
                end_time <- Sys.time()
                run_time <- end_time - start_time
            }
            mape_m <- c(mape_m, mape)
            run_time_m <- c(run_time_m, run_time)
        }
        mape_d_m[match(d_choice, d), ] <- mape_m
        run_time_d_m[match(d_choice, d), ] <- run_time_m
    }

    return(list(mape_d_m, run_time_d_m))
}

# Calculate MAPE For Fixed m
calculate_mape_fixed_m <- function(m, model) {
    mape_n_d <- matrix(, nrow = length(n), ncol = length(d))
    run_time_n_d <- matrix(, nrow = length(n), ncol = length(d))

    for (n_choice in n) {
        mape_d <- c()
        run_time_d <- c()

        for (d_choice in d) {
            rating_matrix_subset <- generate_rating_matrix_subset(
                n_choice, m, d_choice, rating_matrix
            )

            data_frame <- toUserItemRatings(rating_matrix_subset)
            data_frame$userID <- as.numeric(data_frame$userID)
            data_frame$itemID <- as.numeric(data_frame$itemID)
            training_test_sets <- generate_training_test_sets(data_frame)

            if (model == "Linear Model") {
                training_set <- generate_usermean_itemmean(
                    training_test_sets[[1]])
                start_time <- Sys.time()
                mape <- calculate_mape_linear_model(training_set,
                training_test_sets[[2]])
                end_time <- Sys.time()
                run_time <- end_time - start_time
            }
```

```
            if (model == "MF (Reco)") {
                training_test_files <- generate_training_test_files(
                training_test_sets)
                start_time <- Sys.time()
                mape <- calculate_mape_mf_reco(training_test_files[[1]],
                training_test_files[[2]], training_test_sets[[2]]$ratings)
                end_time <- Sys.time()
                run_time <- end_time - start_time
            }

            mape_d <- c(mape_d, mape)
            run_time_d <- c(run_time_d, run_time)
        }
        mape_n_d[match(n_choice, n), ] <- mape_d
        run_time_n_d[match(n_choice, n), ] <- run_time_d
    }

    return(list(mape_n_d, run_time_n_d))
}

# Generate Graph
generate_graph <- function(model, data_source) {
    for (d_choice in d) {
        mape_run_time_m_n <- calculate_mape_fixed_d(d_choice, model)
        generate_graph_for_fixed_d(d_choice, mape_run_time_m_n,
        model, data_source)
    }

    for (n_choice in n) {
        mape_run_time_d_m <- calculate_mape_fixed_n(n_choice, model)
        generate_graph_for_fixed_n(n_choice, mape_run_time_d_m,
        model, data_source)
    }

    for (m_choice in m) {
        mape_run_time_n_d <- calculate_mape_fixed_m(m_choice, model)
        generate_graph_for_fixed_m(m_choice, mape_run_time_n_d,
        model, data_source)
    }
}

# Main
set.seed(123)
data_sources <- list("MovieLens", "InstEval")
```

```
models <- list("Linear Model", "MF (Reco)")
dir.create("./results", showWarnings = FALSE)

for (data_source in data_sources) {
    dir.create(paste0("./results/", data_source), showWarnings = FALSE)

    if (data_source == "MovieLens") {
        data_frame <- ml100kpluscovs
    }
    if (data_source == "InstEval") {
        data_frame <- InstEval[1:30000, ]
        colnames(data_frame)[match("s", colnames(data_frame))] <- "user"
        colnames(data_frame)[match("d", colnames(data_frame))] <- "item"
        colnames(data_frame)[match("sy", colnames(data_frame))] <-
        ↪  "userMean"
        colnames(data_frame)[match("dy", colnames(data_frame))] <-
        ↪  "itemMean"
        colnames(data_frame)[match("y", colnames(data_frame))] <- "rating"
    }

    rating_matrix_info <- generate_fake_rating_matrix(data_frame)
    rating_matrix <- rating_matrix_info[[1]]
    user_mean <- rating_matrix_info[[2]]
    item_mean <- rating_matrix_info[[3]]

    if (data_source == "MovieLens") {
        n <- c(150, 300, 450, 600, 750, length(user_mean))
        m <- c(300, 600, 900, 1200, length(item_mean))
        d <- c(20, 40, 60, 80, 100)
    }
    if (data_source == "InstEval") {
        n <- c(200, 400, 600, 800, 1000, length(user_mean))
        m <- c(200, 400, 600, 800, 1000, length(item_mean))
        d <- c(20, 40, 60, 80, 100)
    }

    for (model in models) {
        dir.create(paste0("./results/", data_source, "/", model),
        showWarnings = FALSE)
        generate_graph(model, data_source)
    }
}
```

B Team Member Contribution

We had several meetings to discuss the general idea of the project. Each of us participated actively in the writing and debugging of the functions. As for the report, we each wrote part of it and each member proofread-ed the entire report.