MovieLens Report

cuisquare

21/04/2021

Contents

1 Introduction

1.1 General Objectives

The MovieLens data set contains the ratings given by users on movies, with additional information on the movie including their genres, title and release year.

The objective of the analysis is to produce rating prediction models that could be used towards a recommendation system for movies, using the MovieLens data set. This would work as follow: A given user will rate a series of movies. Then based on this information, as well as information gathered on other users ratings, the model will provide a prediction for the ratings the user would have given to movies they have not yet watched. The highest rated such unseen movies would constitute the recommendation.

Therefore the objective is to build a model that provides the most accurate prediction of the ratings for movies yet unseen by a user.

1.2 Data preparation and usage

The data set provided was split in a training set (90% of the data) colloquially named *edx* set and *validation* set (10% of the data) without overlap. The *validation* set was then modified to contain only users and movies which are in the training set. The data removed to ensure this, was then added back to the training set.

The validation set thus defined will be used to provide "unseen" data in the sense defined in the General Objectives.

Using the training data, a series of prediction models will be evaluated and tuned. For the purpose of making choices of model type and parameters tuning to best predict ratings, an iterative process will be used comparing predictions to actual ratings. However, the validation data set will not be used for that process. This is because if it was, it would result in overfitting, that is, a model that best fit the data we are considering but might not generalize well to new data. Instead, **only** the *edx* set will be used both for the purpose of providing training data and providing testing data to measure each model performance.

The *edx* set will therefore be further split in 10 randomly selected non overlapping folds. Then a training set will be made up of 9 of the folds, with the 10th remaining fold used as test set to estimate the accuracy of the model. Because the 10 folds are non overlapping it is possible to create 10 different [training;test] pairs, resulting in 10 measures of the model and increasing our confidence that results obtained can be generalised to unseen data.

1.3 Measuring the Model accuracy (RMSE)

The metric used to do that accuracy measurement is the Root Mean Square Error (RMSE) defined as follows:

$$RMSE = \sqrt{\frac{1}{N_{u.i}} \sum_{u,i} (\hat{r_{u,i}} - r_{u,i})^2}$$

where $r_{u,i}$ is the actual rating for movie i, user j and $\hat{r_{u,i}}$ is the corresponding rating estimated using our recommendation system and N the number of user/movie combinations.

The smaller RMSE is, the better our model is likely to be at predicting a user rating for an unseen movie, hence the better our recommendation system should be.

In the case of classic 10 folds cross validation, used to determine the best model, the RMSE used to do so should be the average of the RMSEs obtained for each fold. However for speed of execution we have applied a simplified method where only the "worst fold" was used to tune the model parameters and RMSE calculated for all folds using the same parameters (See section Simplified K Folds cross validation for details)

. Checking results over 10 tolds allows to check the variation of RMSE between data set and validate any improvement in RMSE between models beyond what might be an artefact of the particular (training;test) set pair considerd. This further increases the likelihood that the RMSE obtained be also reached on future unseen data, compared to using a single partition of the *edx* set.

1.4 Meeting the challenge objectives

The challenge sets the target RMSE to be reached on the validation set in increasing order of achievement, as follow :

- RMSE > 0.9 : worth 5 marks
- $0.8655 <$ RMSE < 0.9 : worth 10 marks
- $0.865 <$ RMSE > 0.8655 : worth 15 marks
- $0.8649 <$ RMSE < 0.865 : worth 20 marks
- RMSE < 0.8649 : worth 25 marks

Since it is not possible to know whether this will be reached on the *validation* data set (as it is unseen data), we will consider in this study that a particular target is reached, when it is *likely* to be reached on unseen data. Practically we decide that this be the case when **all RMSE values over randomly selected 10 folds are lower than the particular target with a margin**. This assumption is deemed reasonable considering that:

- the validation data set was randomly generated so it is expected to behave in a similar way as the folds
- with a condition that all folds should meet the target, we should have compensated for the effect of sparse data
- the edx set in its entirety will have more data in it so effects from sparse data for some categories should be less likely
- adding a margin will further reduce the risk the target RMSE be exceeded on unseen data

It is important to note that it is impossible to know for sure how the model will perform on the *validation* set, just estimate how it will perform because it is still possible that the *validation* set is so peculiar that the model does not generalize as well as could be hoped. However considering the above considerations and limitations, this should be unlikely.

2 Methods

2.1 Data exploration

2.1.1 MovieLens Data content

The *edx* set contains a total of 10000054 observations, which have been split as outlined in Section 1.2 into a training set named edx containing 9000055 observations and a validation set named validation containing 999999 observations.

Each observation is made of 6 variables : userId, movieId, rating, timestamp, title, genres.

There are 10677 movies and 69878 users in the edx dataset.

2.1.2 Ratings spread

In Figure 1, one can see that the most given ratings is 4, and that whole star ratings are more common than half star ones.

It also shows that there are no actual ratings above 5 or below 0.5 which therefore appears to be inherent to the grading interface. It implies that any prediction that states a value below 0.5 (resp greater than 5) will be improved by rounding up (resp down) to 0.5 (resp 5), and so this adjustment should be included in the prediction models.

We also see that the range of possible values for actual ratings is *[0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]*. It would therefore be of value to consider whether rounding predictions to the nearest allowed actual rating would improve results on average. This would likely greatly depend on the accuracy of the prediction on average.

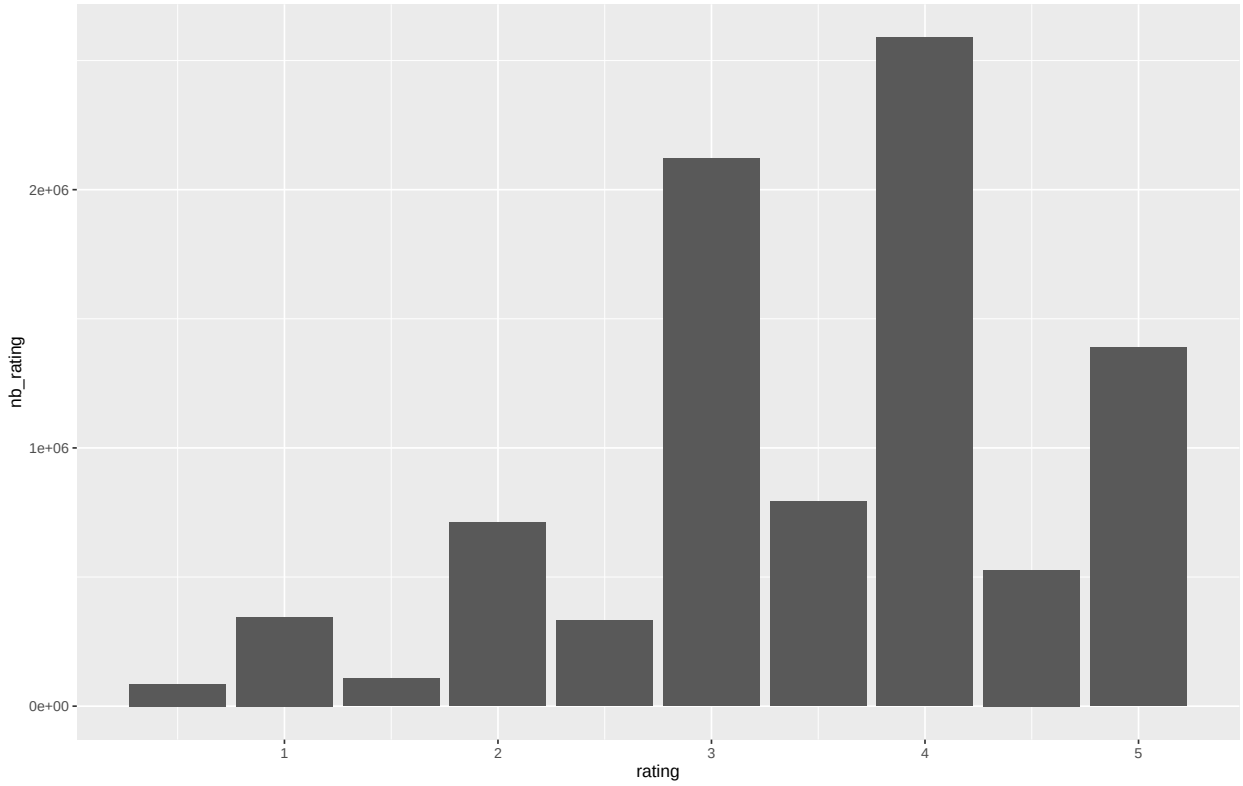

Figure 1: Ratings Spread

Furthermore, it is interesting to consider what the boundaries (minimum and maximum rating) for individual users. This is because if a given user has never given a rating below 2 for instance, it is likely that it will not happen for a new rating by that user. That is, it is likely that for that particular user, a rating of 2 is the minimum it will give to a movie they do not like. This similarly goes for the high ratings. It would be interesting to see whether adjusting for those minimum and maximum ratings for each individual users treating them effectively as new lower and upper boundaries improves predictions on average.

2.1.3 Genres spread

In Figure 2, one can see that there is a large number of movies for each individual genre, with the most represented genre being Drama followed by Comedy.

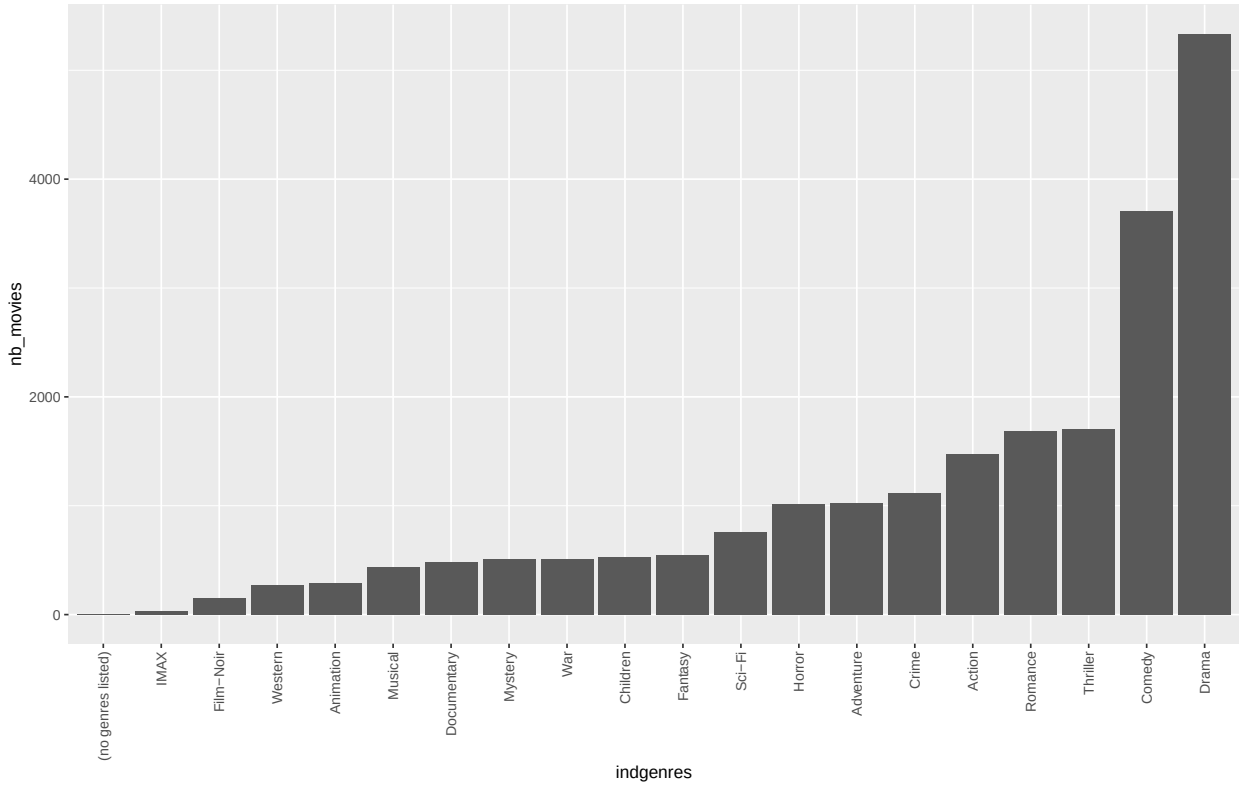

Figure 2: Nb Movies per Individual Genre

2.2 Data cleaning

2.2.1 Release year extraction

The movie titles contain the release year within brackets at the end of the string, which can be extracted using regular expressions with the pattern "\((\d{4})\)$" which matches a series of 4 digits within brackets immediately preceding the end of the string.

2.2.2 Individual genres extraction

The data has a genres field which is a string made of the concatenation of individual genres separated by the symbol "|". To extract the effect of each genre, the genres field is split using the unnest function of the tidyr package.

2.3 Modelling approach

2.3.1 General Model

We are considering the rating for user u to be modeled by the sum of their usual rating (average rating for user u) plus additional effects likely to make that rating vary away from that average. The effects will be evaluated and the rest of the variation will be considered to be random variation.

We are considering the rating for movie i and user u to be a result of the average obtained for user u on other movies and for movie i by others with the remainder of the variation explained by random variation.

The model can therefore be written as follows:

$$r_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

where μ is the average rating across all ratings in the training set, b_i is the effect of movie i and b_u is the effect of user u.

We cannot know the exact values for the parameters but we can estimate the value that will minimise the RMSE. Since in our model we assume that $\epsilon_{u,i}$ is a random error centred around 0, the value that minimise the RMSE will be the average of the non random terms.

The effects are obtained by looking at the deviation from the overall mean when considering the different movies. We calculate μ_i the averages for each movie i the subtract the overall average μ giving b_i:

$$\hat{\mu_i} = \frac{1}{N_u} \sum_u r_{i,u}$$

$$\hat{b_i} = \hat{\mu_i} - \hat{\mu} = \frac{1}{N_u} \sum_u (r_{i,u} - \hat{\mu})$$

We then do the same to obtain b_u:

$$\hat{\mu_u} = \frac{1}{N_i} \sum_i r_{u,i}$$

$$\hat{b_u} = \hat{\mu_u} - \hat{\mu_i} = \hat{\mu_u} - \hat{\mu} - \hat{b_i} = \frac{1}{N_i} \sum_i (r_{u,i} - \hat{\mu} - \hat{b_i})$$

This is one example of effects, but others can be considered such as the genres, or the release year or the time of the day a rating was given, or day of the week, or period of the year.

2.3.2 Regularisation

Regularisation can also be applied to avoid over fitting the solution in case of a category with a small number of observations, for instance movies with small number of ratings. The average rating for a given movie (resp user) is divided not by the number of users (resp movies) but by this number with a fixed weighting added. This will have the effect to reduce the end value of the deviation to a greater extend for groups with a small number of observations so that overfitting (overtraining) can be avoided and the model generalise better - resulting in better predictions.

Practically we are now trying to minimise the following equation:

$$MSE_\lambda = \frac{1}{N_{u,i}} \sum_{u,i} (r_{u,i} - \hat{\mu} - \hat{b_i} - \hat{b_u})^2 + \lambda \left(\sum_i b_i^2 + \sum_u b_u^2 \right)$$

And it can be proven that the minimisation equations becomes:

$$\hat{b_i} = \frac{1}{N_i + \lambda} \sum_u (r_{u,i} - \hat{\mu})_i$$

$$\hat{b_u} = \frac{1}{N_u + \lambda} \sum_i (r_{u,i} - \hat{\mu} - \hat{b_i})_u$$

One can see from the equations that in general the effect is smaller than it would be without the parameter λ. In the case where the number of observations is small,that effect reduction will be significant; in the case where the number of observation is large, the term $N_u + \lambda$ will be close to N_u.

Supposing we stick to the model with only movie and user effect, we assume that ratings for movie i, user u is:

$$r_{u,i} = \mu + b_i + b_u + \epsilon_{u,i}$$

Which means our estimate of the rating for user u and movie i will be:

$$\hat{r_{u,i}} = \hat{\mu} + \hat{b_i} + \hat{b_u}$$

Each time we decide on a set of effects, we assume that any variation further than those resulting from the effects that have been included thus far (for instance, what the movie is and which user is rating it), can be explained by random variation.

One matter of consideration is whether one should use a single lambda for all effects, or if there is value in having different lambdas for the different effects. In the the run of the models, both cases will be considered and the choice giving the lowest RMSE will be picked.

2.3.3 Further effects

We can further refine the model by considering the remaining variation can be explained by other factors and is not in fact random.

Further effects can either be applied over the whole set of users to derive an effect that is supposedly common to all users, or for each user individually. The second way will give more accurate effects for an individual user, however only provided that there were enough observations with that particular effect represented, to allow for an accurate effect to be determined. In the worst case scenario there might be no observation for a given individual user for that effect, in which case an effect common to all the uers would have been a better estimate, because of the lack of data.

2.3.3.1 movies genres Genres will influence the rating of a movie based on users preference. There are a few ways that genres effect can be modeled. A first model could consider the combination of genres as presented in the data, as a genre itself. This however mean that combinations of genres very similar such as Drama|Comedy|Thriller and Drama|Comedy|Action will be treated completely independently, when most likely there is commonality in their effect. Furthermore it will lead to data that is more sparse as the particular combination might not have a lot of observations.

$$\hat{b_g} = \frac{1}{N_g + \lambda_g} \sum_i (r_{u,i} - \hat{\mu} - \hat{b_i} - \hat{b_u})$$

$$\hat{r_{u,i}} = \hat{\mu} + \hat{b_i} + \hat{b_u} + \hat{b_g}$$

An improvement on that model is to separate the individual genres and consider their effect. However by doing so, a given movie with several genres will need to receive an aggregate of the effects of each genre. A proposed model would be to consider for a given movie to get a genres effect that is the average of the effect of each individual genre.

Because the genres are likely to have user specific effect, we extract one such effect per existing user. To simplify the computation, we use the same λ for all the individual genres.

$$\hat{b_{u,indg}} = \frac{1}{N_{indg} + \lambda_{indg}} \sum_i (r_{u,i} - \hat{\mu} - \hat{b_i} - \hat{b_u})_{u,indg}$$

$$\hat{b_{u,g}} = mean(\hat{b_{u,indg}})$$

$$\hat{r_{u,i}} = \hat{\mu} + \hat{b_i} + \hat{b_u} + \hat{b_{u,g}}$$

2.3.3.2 movies release date As a movie has a single release date the effect is straightforwardly applied by taking the average over all movies of the same release date. Again, because the year preference is likely to be a personal preference, the effect is extracted for each user.

$$\hat{b_{u,ry}} = \frac{1}{N_{ry} + \lambda_{ry}} \sum_i (r_{u,i} - \hat{\mu} - \hat{b_i} - \hat{b_u})_{u,ry}$$

2.3.3.3 rating date We can also consider the effect of when the rating was given. This will reflect whether the average rating given by a user varies with time. Again, because a time variation in rating is likely to be specific to each user, the effect is extracted for each user.

$$\hat{b_{u,t}} = \frac{1}{N_t + \lambda_t} \sum_i (r_{u,i} - \hat{\mu} - \hat{b_i} - \hat{b_u})_{u,t}$$

2.4 Numerical methods

2.4.1 Simplified K Folds cross validation

Because we try to minimise the RMSE with lambda as a variable, in K folds cross validation we normally would modify the RMSE function to be the average or median RMSE over the K folds using that lambda parameter. This means we have K times more prediction function evaluation to carry out. Furthermore in some of the models there might be value in using different lambdas for each effects. This can be numerically demanding resulting in a very long run time.

For this reason, a simpler method was chosen, which is to determine the minimising lambda only for one of the folds, then apply this lambda to all folds to obtain the 10 RMSEs, as a way to check that the RMSE did overall vary in the same direction across models.

The choice for the particular fold to use was to pick the fold that gave the highest RMSE values among all folds, for the model that on average over the 10 folds gave the lowest RMSE over the models considered thus far. In other words, the "worst" fold for the "best" model. This was because it was considered that the likelihood that the lambda chosen has an impact was considered greater in the cases where the RMSE would be the highest. Therefore it could be supposed that this might improve the most results for data yet unseen.

Because of this simplification it will be necessary to check whether indeed each model bringing improvement in the RMSE over the worst fold, will also bring similar improvement on all other folds.

2.4.2 Function minimisation : Golden section search

Even with the simplification of determining only the lambda(s) minimising the worst fold, it still remains that the determination of lambda can be long if a specific precision is to be reached. A method was chosen that allowed to get to the minimum RMSE for a given model within a chosen precision with as little function evaluations as possible.

The basic method is to pick a minimum and maximum lambda and then pick 2 lambda values between those first two guesses to find lambda values that:

- yield a lower RMSE than both the minimum and maximum lambda
- provide a new interval between which to search for new lambdas further minimising the function
- has a consistent lambda spacing to optimise the search and reduce function evaluations

The method chosen was to apply the golden section search.

For a function f of the monovariate variable x, for which we are searching for a minimum between two values Xmin and Xmax, we pick two further x values that are spaced so that the golden ratio is applied between the two inner and outer values. See Figure 3 [1] for a diagram of the search.

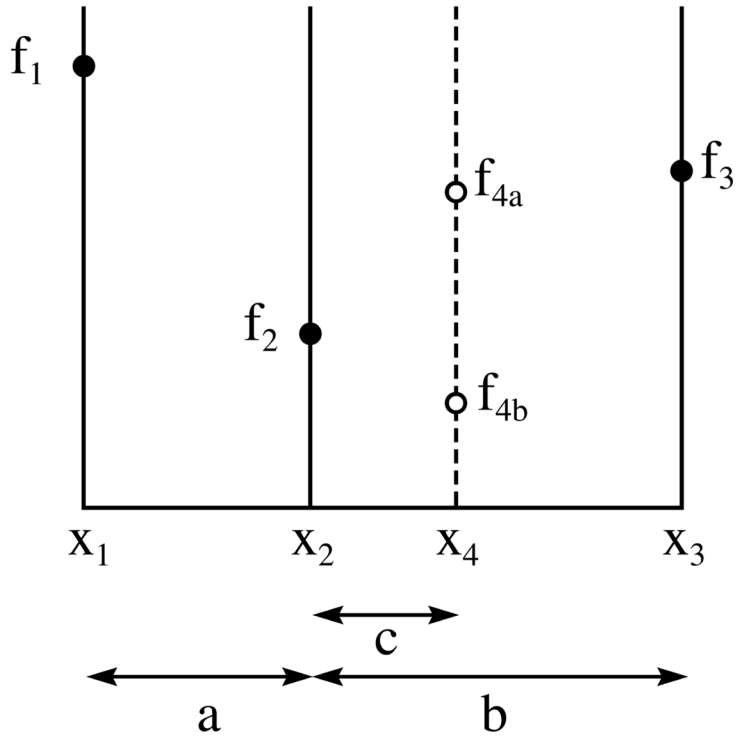

Figure 3: Golden Section Search

Practically we pick the x values so that the ratios are:

$$\varphi = \frac{c}{a} = \frac{c}{b-c} = \frac{a}{b}$$

It can be proven that this is the case if φ is the golden ratio defined as:

$$\varphi = \frac{1+\sqrt{5}}{2}$$

Once the values of x_2 and x_4 are calculated, we evaluate f at those points. Then, if f4 is greater (resp lower) than f2 , the new search interval becomes $[x_1; x_4]$ (resp $[x_2; x_3]$). We then select the new x value in the $[x_1; x_2]$ (resp $[x_2; x_3]$) interval that the golden ratio is again met.

More details on this method can be found at the following wikipedia page Golden Section Search - Wikipedia and the implementation of the function in R is in the HELPER FUNCTIONS DEFINITION section of the RProf_Capstone_MovieLens R script.

Example of the method used in the context of this study can be found in section RMSE function Minimum Search.

[1] (copyright CC BY-SA 3.0 PAR)

3 Results

3.1 Simplified Cross Validation Results

3.1.1 Summary

The best model considered is one that incorporates the following effects: *movie+user+indgenres+releaseyear+timestamp reg* achieving a median RMSE of *0.839424* with a maximum RMSE of **0.8402347** obtained on Fold01.

The best model without regularisation incorporates the following effects: movie+user+indgenres achieving a median RMSE of *0.85074* with a maximum RMSE of **0.8514515** obtained on Fold01.

Since every fold out of the 10 randomly generated Folds on the training data beat the lowest target RMSE = 0.8649 we achieved the objective set in section Meeting the challenge objectives for a reasonable expectation to beat the target on future unseen data for both of those models.

The following models also achieved the objective set to beat the lowest target RMSE:

- movie+user+genres reg (max_RMSE = **0.85732**)
- movie+user+indgenres reg (max_RMSE = **0.84956**)
- movie+user+indgenres+releaseyear reg (max_RMSE = **0.84646**)

The full simplified cross validation results for the models considered can be found in Table 2 and Table 3, ordered by descending mean RMSE (increasing mean prediction accuracy) over the 10 folds randomly generated, and located in the Full Simplified Cross Validation Results section of the Appendix.

3.1.2 Visualisations

Figure 4 is the plot of all models RMSE for all folds with boxplot superimposed. The target RMSE are drawn as horizontal lines with the following colours:

- RMSE = 0.9 : red
- RMSE = 0.8655 : orange
- RMSE = 0.865 : blue
- RMSE = 0.8649 : green

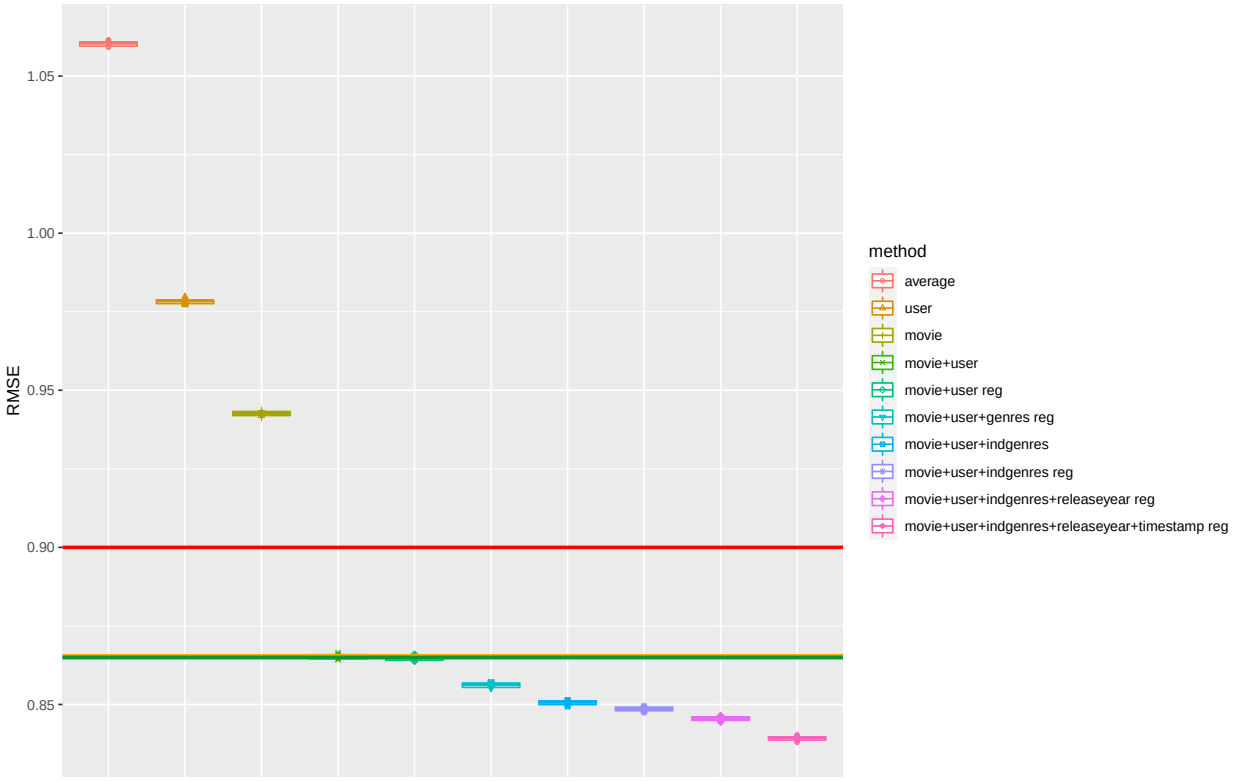

Figure 4: RMSE variation with models

Figure 5 is the same plot, with only the models with RMSE below 0.88. It shows clearly that a model just including movie+users even with regularisation does not allow to achieve the RMSE target of 0.8649 for all folds, so makes the case to have considered more complex models with additional effects.

It also shows that the lowest target RMSE of *0.8649* is beaten with a significant margin by 5 of the models.

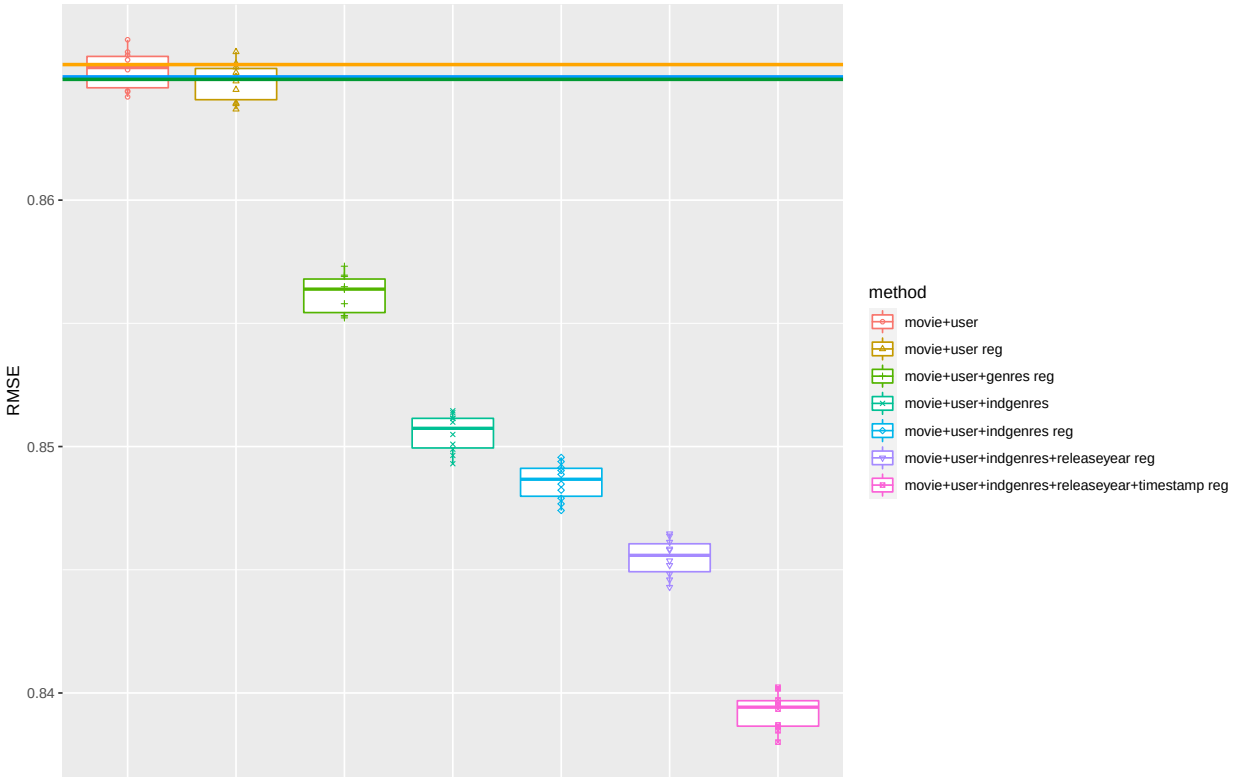

Figure 5: RMSE variation with models (below 0.88 RMSE only)

It also provides verification that the simplified cross validation method described in section Simplified K Folds cross validation is valid indeed for each model, all 10 folds are within roughly the same RMSE interval with a standard deviation of around 0.00075 and follow the same variation in RMSE from model to model. It is even more clear when drawing each fold variation across models as a line plot in Figure 6 that tuning on the worst fold does improve the results on all folds in a very similar manner.

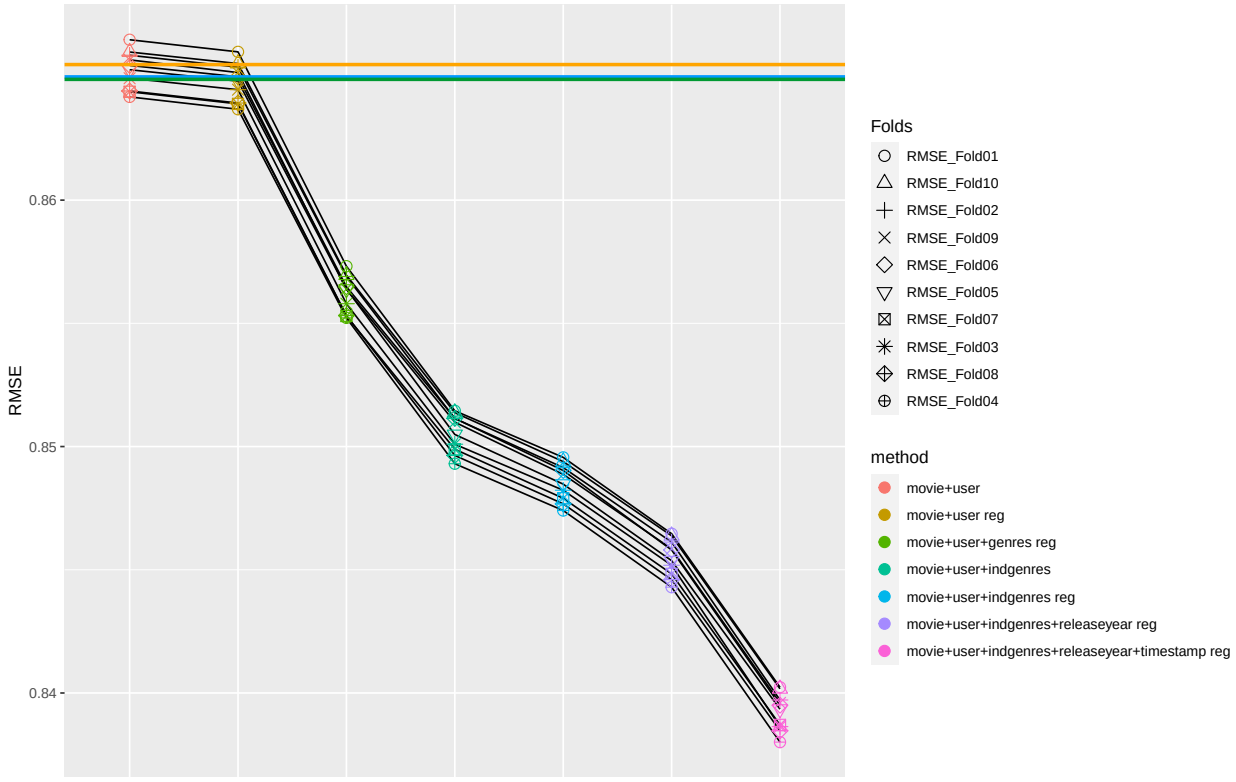

Figure 6: RMSE variation for each fold (below 0.88 RMSE only)

3.2 Final model prediction on validation set

Using the best models derived, it is now possible to calculate the RMSE on the unseen data contained in the *validation* set. To do so, the model is trained one last time on the whole *edx* (training) set, using the lambda parameter previously derived from simplified cross validation on the worst fold (fold01) (with verification on the other 9 folds), then a prediction is made on the *validation* set for which we calculate the RMSE.

Doing so yields the following result:

- **0.8380426** for the *movie+user+indgenres+releaseyear+timestamp reg* model
- **0.8492771** for the *movie+user+indgenres* model

Both models do beat the lowest target RMSE of *0.8649* by a significant margin.

Also we observe that both RMSE values are lower than the RMSE obtained on the "worst fold" during the 10 folds simplified cross validation.

This reinforces the idea that the proposed method to use the worst fold to provide a conservative estimate of what to expect on unseen data and to tune the regularised models parameters on that worst fold, was reasonable.

The prediction on the validation set was was produced *3.24* times slower for the regularised model compared to the non regularised model. To this should be added the time required to tune the lambda parameters of the regularised model during simplified cross validation. This highlights that depending on the target RMSE choices the non regularised model appears like a good choice if speed of execution is a factor.

4 Conclusion and Future Work

Using a model that incorporated the effects of movie, user, date of rating, individual genres and release year to account for the variation in ratings, it was possible to reach predictions in movie ratings for unseen data with a RMSE below 0.85, beating the lowest RMSE target of 0.8649.

The exploration of models has shown that regularisation brings improvement in predictions. Furthermore it was observed that using different lambda parameters for the different effects can bring further improvements in some cases. The effect of rounding predictions to the nearest allowed values ws considered but did not bring improvements. However considering the boundaries of the ratings for each user, it was possible to reduce the RMSE. See section Refinements brought to Models for more details.

Further improvement might be possible by using full cross validation whereby the lambda values are obtained by minimising a modified RMSE function that gives the average of the RMSE across the folds. This was not done, in order to shorten the run times of the training of the models (since each modified RMSE evaluation would require a number of computation K times higher, if K is the number of folds).

Further models could also be used to reach further reduction in RMSE, such as matrix factorisation applied on the residuals of the best model.

It would also be of value to consider what happens when sparse data is not removed from the training set. That is, when we allow for data in the validation set to contain both movies and users which are not present in the training set. This would be a more general situation closer to reality as it would correspond to new movies added to the library or new users joining the system.

5 Appendix

5.1 Refinements brought to Models

5.1.1 Rating Rounding / Adjustments

As discussed in section Ratings spread, The possible values for user given ratings are discrete and the possible values are: *[0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5]*.

There was one more applied adjustment to the predictions based on the allowed interval that was an immediate win. Indeed for models that have more than one effect, the prediction can end up being outside of the possible interval for actual rating. Taking the first model with two effects, movie+users, we find that for the worst fold, the number of predictions which are lower than 0.5 or higher than 5 is *2161* out of *899991*. We therefore adjust any rating below 0.5 to 0.5 and any rating above 5 to 5. Adjusting for global boundary reduced the RMSE from the original *0.8667928* to *0.8665966*. We then also checked that applying user boundary after global boundary adjustment gave a further reduction in RMSE, which was the case as we reached a RMSE of *0.8665076*. This was done because in cases where a user has not rated movies yet there will be no rating boundary information available.

We considered the effect of rounding the results to the nearest allowed ratings but this did not bring improvements. The movie model without any rounding yielded a RMSE of *0.9446996* while the rounded model yielded a higher RMSE of *0.9448821* when applying rounding only to the nearest whole rating, for users who had only given whole ratings, and only for predictions that were within 0.1 of a whole rating.

Next we considered the effect of user boundaries in their ratings. We rounded all predictions down (resp up) to the user maximum (resp minimum) rating given thus far if they were found to be greater (resp lower) than that user maximum (resp minimum) rating. This was based on the assumption that each user might have their idea of what is a maximum and minimum rating, keeping to this rather than the full range. By doing so, we yielded a *lower* RMSE of *0.9435589*. As a result we applied this adjustment for all further models without specifying it in the model descriptions for conciseness.

We also applied adjustment on the predictions based on the allowed interval that was an immediate win. Indeed for models that have more than one effect, the prediction can end up being outside of the possible interval for actual rating. Taking the first model with two effects, movie+users, we found that for the worst fold, the number of predictions which are lower than 0.5 or higher than 5 is *2161* out of *899991*. We therefore adjusted any rating below 0.5 to 0.5 and any rating above 5 to 5. Adjusting for global boundary reduced the RMSE from the original *0.8667928* to *0.8665966*. We then also checked that applying user boundary after global boundary adjustment gave a further reduction in RMSE, which was the case as we reached a RMSE of *0.8665076*. This was done because in cases where a user has not rated movies yet there will be no rating boundary information available.

Considering the above, for all models with more than one effect, we have applied global then user boundary adjustment, without specifying it in the model descriptions for conciseness.

5.1.2 Multiple Lambda vs Single Lambda Regularisation

When considering models that were regularised, we have considered two cases:

- a single lambda parameter was used across all effects
- one specific lambda parameter for each new effect when adding a new effect

Table 1: Single vs Multiple Lambda RMSE Comparison

method	SingleLambdaRMSE	MultipleLambdaRMSE	BestModel
movie+user	0.8660161	0.8660269	Single Lambda
movie+user+genres	0.8573178	0.8576965	Single Lambda
movie+user+indgenres	0.8496666	0.8495601	Multiple Lambdas
movie+user+indgenres+releaseyear	0.8495612	0.8464639	Multiple Lambdas
movie+user+indgenres+releaseyear+timestamp	0.8437321	0.8402347	Multiple Lambdas

Results were found to be better with a single Lambda for the following models :

- movie+user
- movie+user+genres

Results were found to be better with multiple lambdas for the following models:

- movie+user+indgenres
- movie+user+indgenres+releaseyear
- movie+user+indgenres+releaseyear+timestamp

In all regularised models, we reported the version (single or multiple lambdas) that gave the lowest RMSE.

5.2 Full Simplified Cross Validation Results

Similarly to the Visualisations section, we have colour coded the RMSE results, as follow :

- RMSE > 0.9 : black
- $0.8655 <$ RMSE < 0.9 : red
- $0.865 <$ RMSE > 0.8655 : orange
- $0.8649 <$ RMSE < 0.865 : blue
- RMSE < 0.8649 : green

Table 2: 10 Folds RMSE results

method	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
user	0.97978	0.97868	0.97758	0.97721	0.97851	0.97932	0.97758	0.97717	0.97845	0.9785
movie	0.94356	0.94326	0.94188	0.94218	0.94276	0.94261	0.94146	0.94132	0.94324	0.94321
movie+user	0.86651	0.86588	0.86495	0.86419	0.86529	0.86547	0.8644	0.86443	0.86569	0.86601
movie+user reg	0.86602	0.8654	0.86448	0.86369	0.86483	0.865	0.8639	0.86394	0.86518	0.86554
movie+user+genres reg	0.85732	0.8569	0.8558	0.85522	0.85639	0.85649	0.8553	0.85532	0.85639	0.85695
movie+user+indgenres	0.85145	0.85114	0.8501	0.8493	0.85049	0.85115	0.84989	0.84964	0.85099	0.85137
movie+user+indgenres reg	0.84956	0.84915	0.84823	0.8474	0.84847	0.84903	0.8479	0.84768	0.84887	0.84939
movie+user+indgenres+releaseyear reg	0.84646	0.84612	0.84518	0.84429	0.84537	0.8458	0.84483	0.8446	0.84585	0.84638
movie+user+indgenres+releaseyear+timestamp reg	0.84023	0.83972	0.83864	0.83801	0.83935	0.8395	0.83871	0.83847	0.8396	0.84017
average	1.06156	1.06126	1.05958	1.05956	1.06053	1.06094	1.0594	1.05922	1.06049	1.06076

Table 3: 10 Folds RMSE results Summary

method	mean	median	max	sd
average	1.06033	1.06051	1.06156	0.00083
user	0.97828	0.97848	0.97978	0.00088
movie	0.94255	0.94269	0.94356	0.00080
movie+user	0.86528	0.86538	0.86651	0.00078
movie+user reg	0.8648	0.86492	0.86602	0.00078
movie+user+genres reg	0.85621	0.85639	0.85732	0.00076
movie+user+indgenres	0.85055	0.85074	0.85145	0.00078
movie+user+indgenres reg	0.84857	0.84867	0.84956	0.00075
movie+user+indgenres+releaseyear reg	0.84549	0.84559	0.84646	0.00076
movie+user+indgenres+releaseyear+timestamp reg	0.83924	0.83942	0.84023	0.00075

5.3 RMSE function Minimum Search

5.3.1 Extensive Search

We compare the minimum RMSE search methods. The first approach is to extensively search for the minimizing lambda by calculating the RMSE for a series of lambda until a minimum is reached. For the *movie reg* model, we get the following search graph.

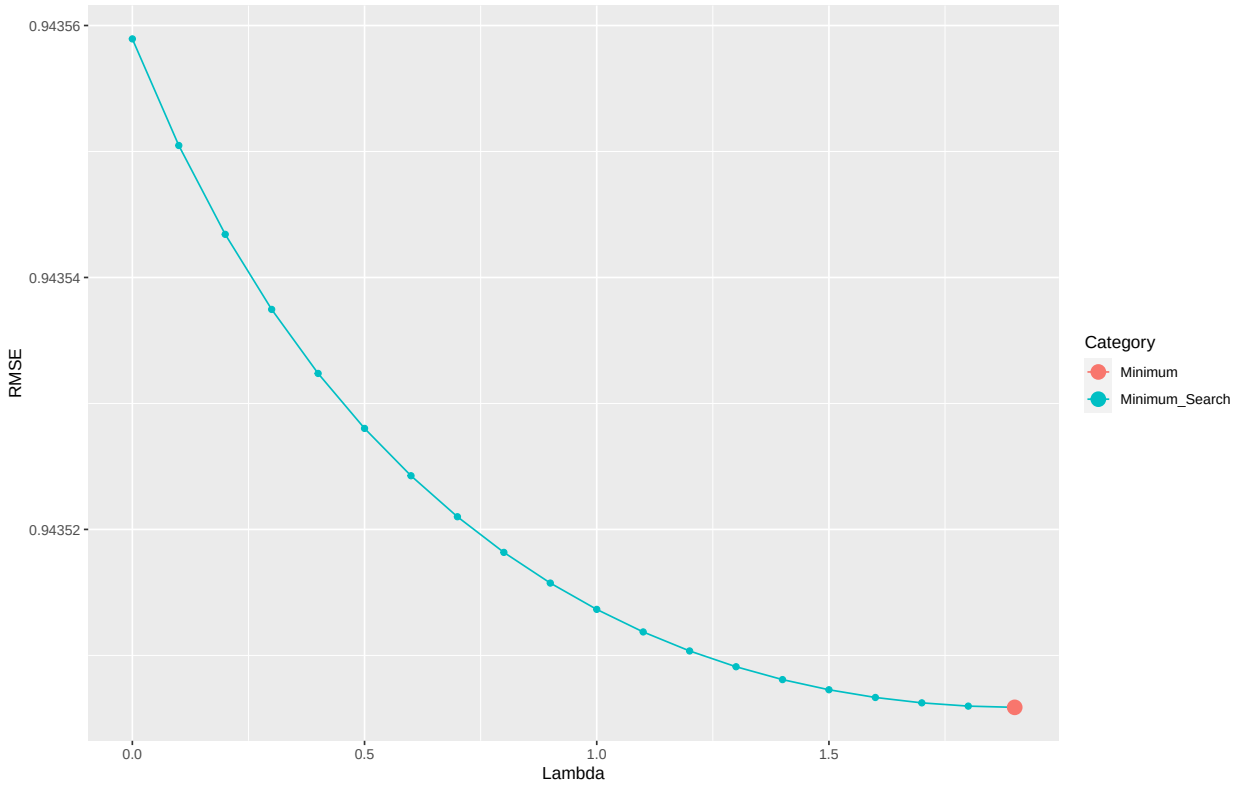

Figure 7: Extensive Minimum Search

5.3.2 Golden Section Search

In the golden search method, we start with 4 lambda values with the minimum and maximum being the start and end of the search interval, and the two middle lambda values picked so that the golden ratio is met.

The advantage is that each further step in the search only requires a single RMSE function evaluation. Overall, we reach a minimum after *6* function evaluations.

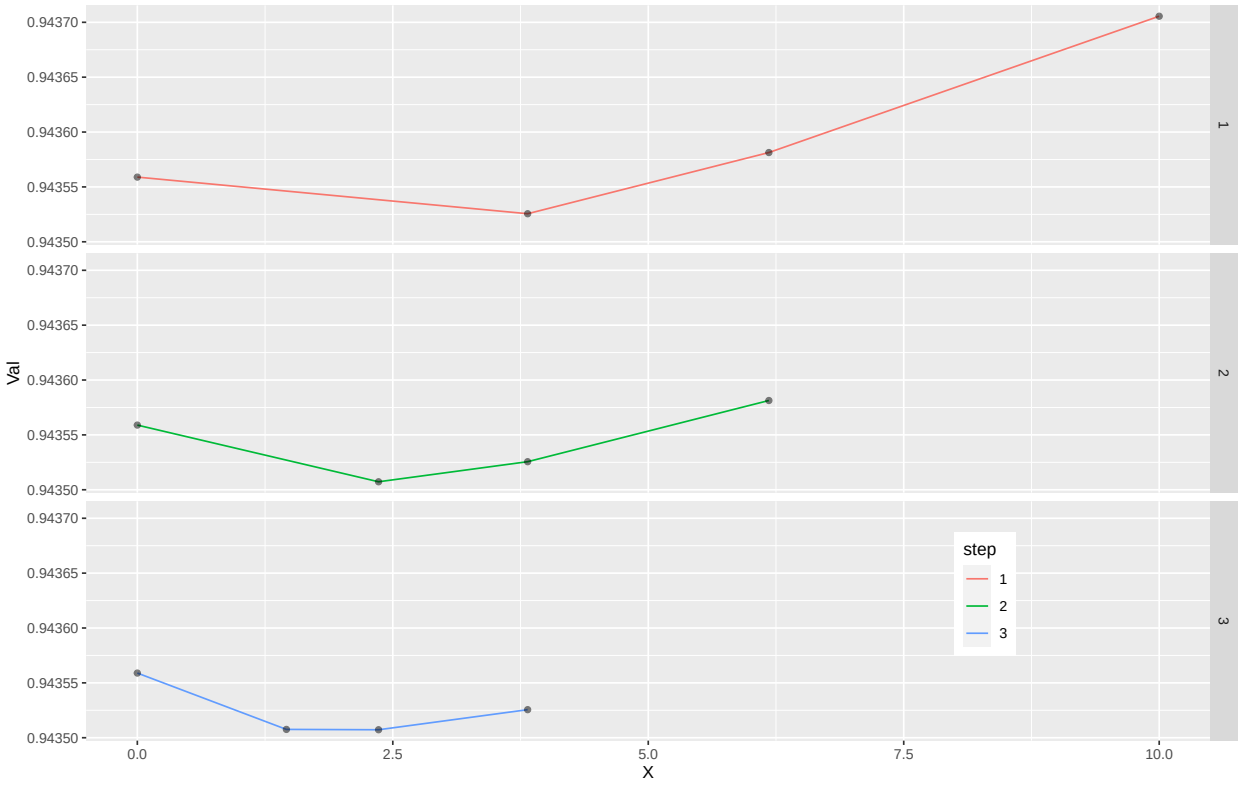

Figure 8: Golden Seaction Search Method - Steps Faceted

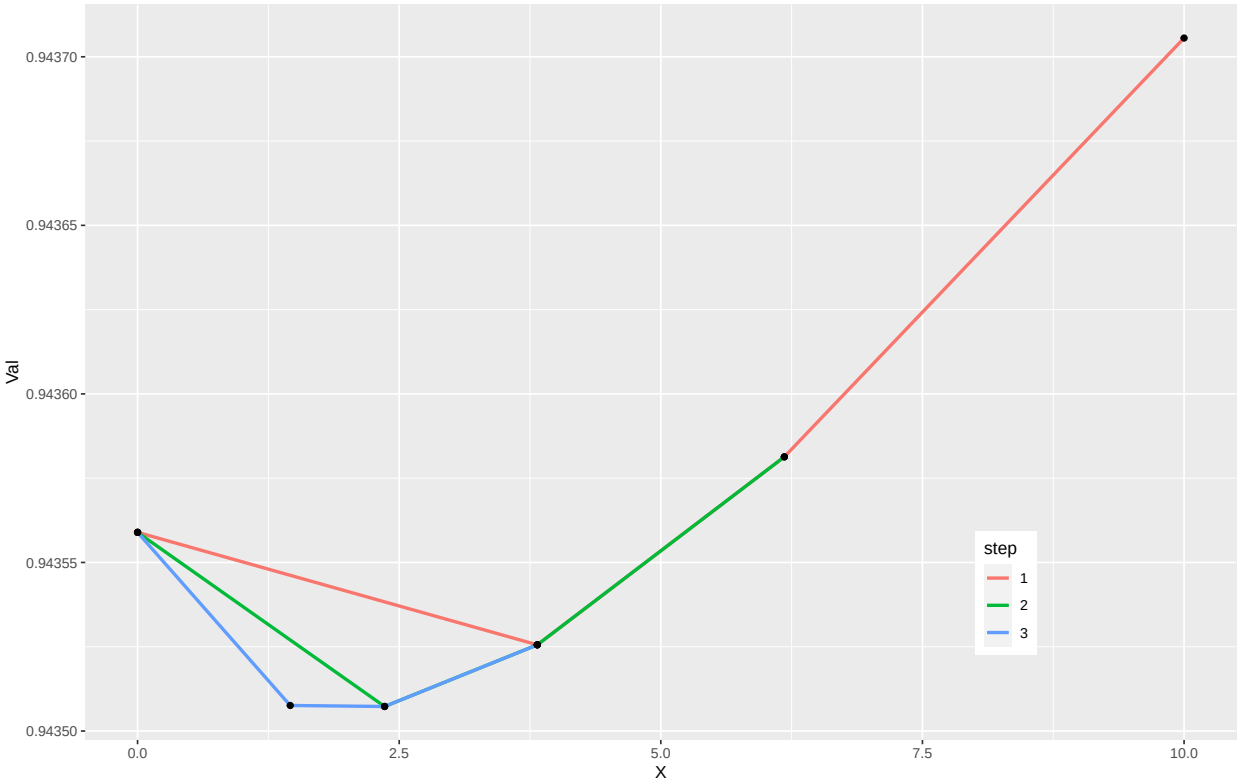

Figure 9: Golden Seaction Search Method - Steps Superimposed

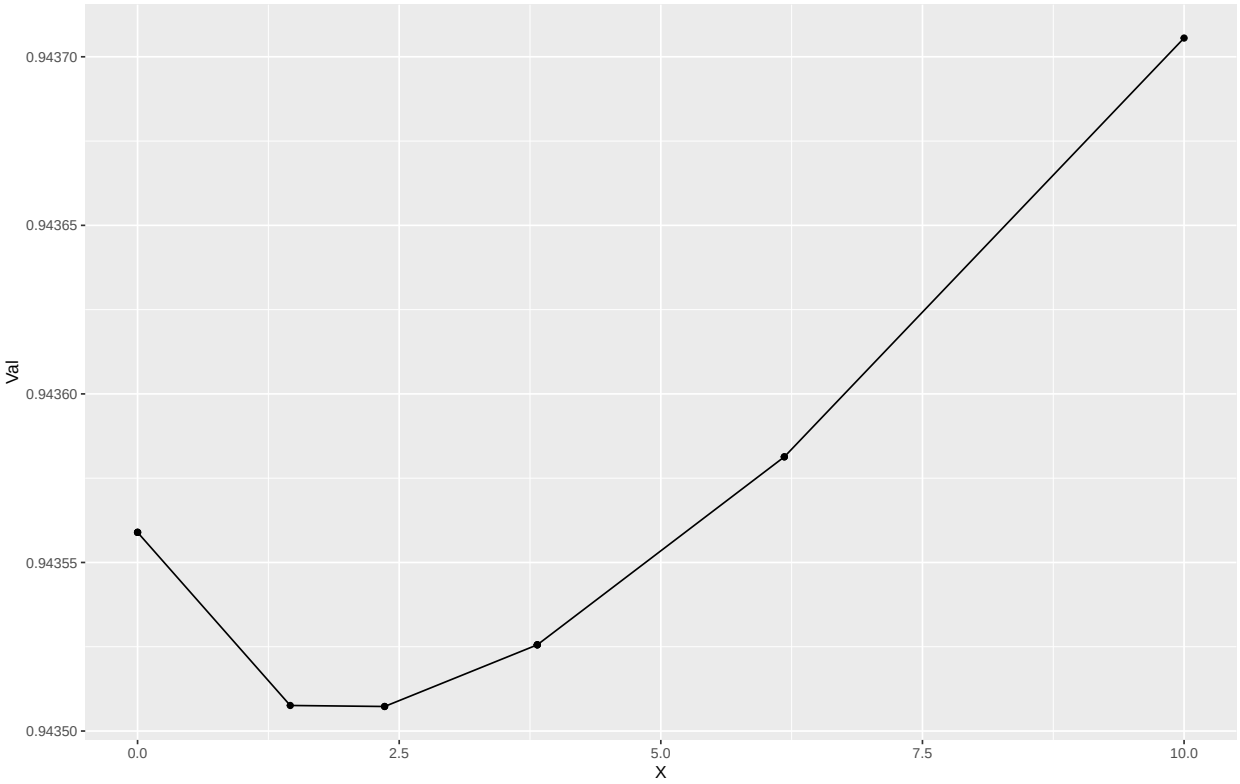

Figure 10: Golden Seaction Search Method - All Values

5.3.3 Search method comparison

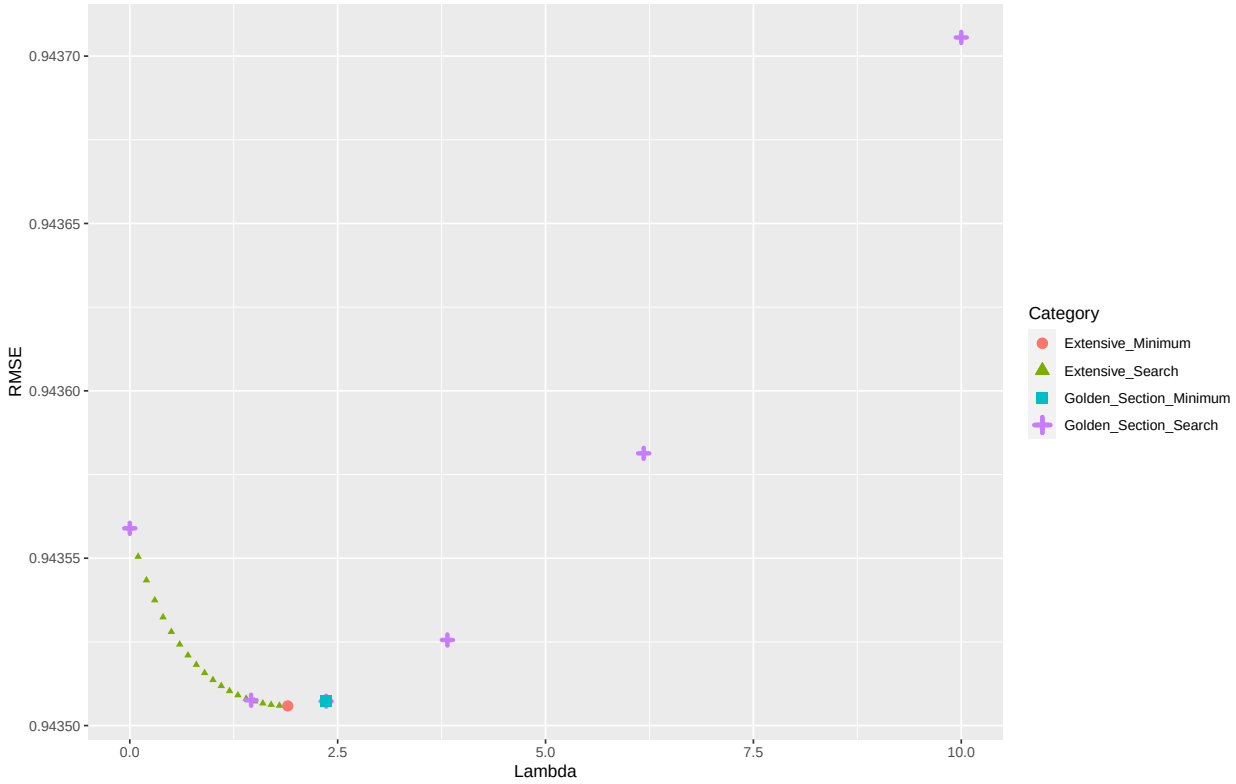

Figure 11: Search Method Comparison

In figure 11 the advantage of using the Golden Section search is quite clear since with only *6* RMSE function evaluations we do reach a minimum RMSE of *0.94351* which is virtually identical to the one found for the Extensive Search method (*0.94351*), while that latter method requires *20* RMSE function evaluations (*3.33* as many evaluations).

5.3.4 Golden Section Search Limitations : search interval

One can see from the definition of the method that for it to progress it requires the values at XLeft and XRight to be such that there is a minimum found at step number 1 of the search. But this might not be the case, depending on the search interval chosen! Figure 12 below is an example of such a case, using the *movie reg* model. By choosing the search interval to be *[0-50]* step 1 picks values of lambda which do not show a minimum.

To alleviate this issue, the function was improved to allow for further attempts finding a minimum in case of initial failure. If a function is found to be absolutely increasing at step 1, then further attempt is made to find a minimum between XMin and XLeft, since this is the only interval where such a minimum will be if it exists. It is then possible to find the minimum even if the initial interval chosen is quite large provided that enough attempts are made. Figure 12 below shows that allowing two further such attempts results in finding the minimum.

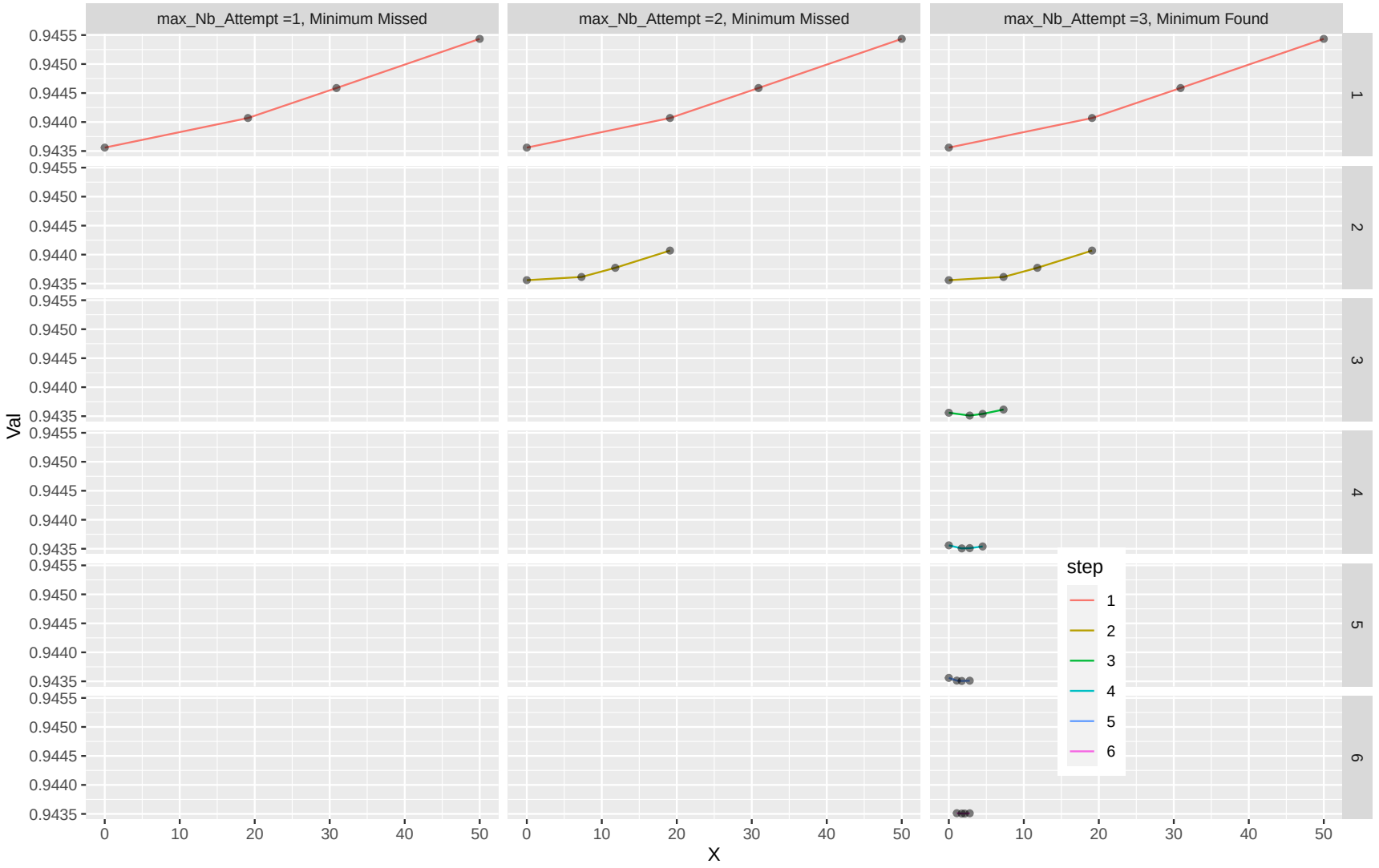

Figure 12: Larger Interval Search Limitation and Resolution