Predicting DVD Unit Sales Based on Popularity Measures

UCSC Silicon Valley Extension
2612 Introduction to Machine Learning and Data Mining

Andrew Dempsey

Table of Contents

Summary

From wikipedia: http://en.wikipedia.org/wiki/Demand_forecasting

> ***Demand forecasting*** *is the activity of estimating the quantity of a product or service that consumers will purchase. Demand forecasting involves techniques including both informal methods, such as educated guesses, and quantitative methods, such as the use of historical sales data or current data from test markets. Demand forecasting may be used in making pricing decisions, in assessing future capacity requirements, or in making decisions on whether to enter a new market.*

The ability to predict the retail demand of a product prior to its availability enables a retailer to purchase inventory in a more efficient manner. Hopefully avoiding both over buying (leading to discounted sales) and under buying (leading to shortages and unhappy customers). More efficient purchasing, leads to more efficient sales and a maximization of profit.

Just as importantly, the ability to predict of DVD sales units would help the studios, manufacturers and distributors plan for better production and distribution of the physical discs.
http://articles.latimes.com/2009/may/19/entertainment/et-bigpicture19

The idea is, is that by taking box office performance data (especially opening weekend, a measure that is available much earlier than total US or worldwide box office receipts) and multiple sources of ratings (rottentomatoes.com, new york times etc), it will be possible to predict the DVD demand in retail sales units for up to the first 8 week window of sales (from DVD street date).

Assuming a reasonable level of accuracy is achievable, a real world implementation of this approach would potentially:

- Include internal corporate data to fine tune the algorithm to the specific needs of the company
- Be used to manage initial inventory purchases for retailers (by their 'buyers')
- Subsequent restock purchasing would require a different type of algorithm due to the decay in demand over a relatively short period of time (e.g. the demand for a particular DVD title after 6 months, is significantly lower that at time of release and can be affected by other factors such as seasonality, Oscar nominations, deaths of star actors etc)
- Be used by studios and manufactures to decide how many units to manufacture

For the purposes of this project, I shall only be considering the 2009-2011 time frame of theatrical releases.

Collecting Data

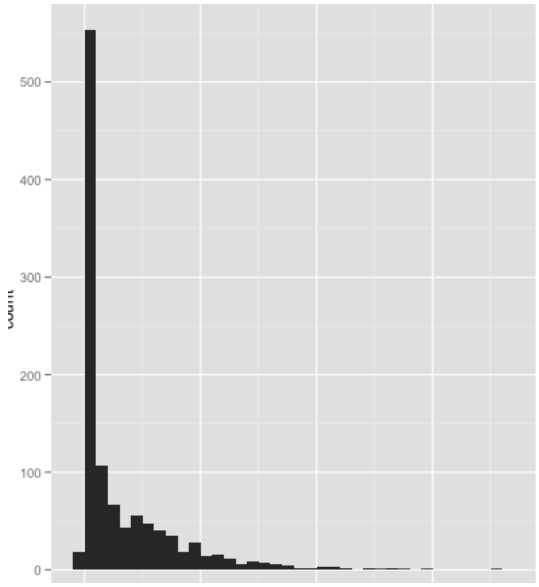
All efforts were made in the allotted time frame, to obtain data for this project from the public domain.

DVD Sales Data

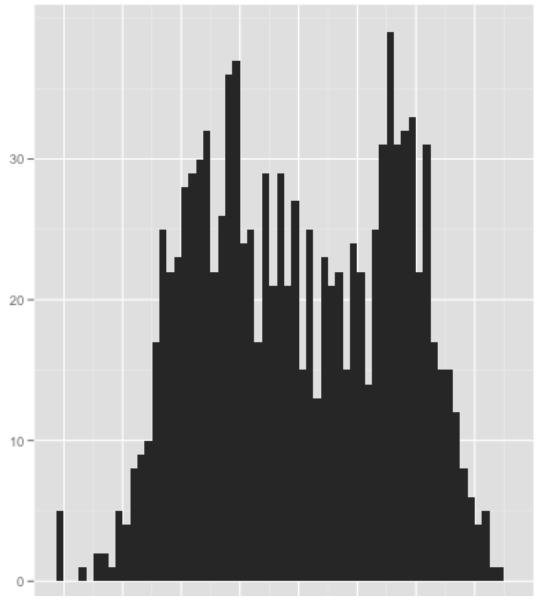
Unfortunately, a reliable public domain source for DVD sales units could not be found. The closest web site, The-Numbers.com, offers only patchy coverage of movie titles that had their theatrical release during the 2009-2011 time frame.

Therefore, a third party commercial source was used to obtain this data, which cannot be shared directly with the class. However, a suitable transformation of the data from this source was applied, for use in the project.

The original raw data has the following distribution, with a long tail to the right. Indicating that a small number of Hollywood blockbusters, really skew the data.



The transformation I applied, produces a different pattern, where the long tail to the right is compressed and the tall peak on the left is expanded to give a better 'separation' to the main field.



Before applying this transformation, I performed some grouping of records in order to ignore DVD format (standard, 3D, blu ray etc.) and special editions (directors cut, multi-packs etc), so that I would have a single sales volume figure per movie.

Box Office Receipts

BoxOfficeMojo.com is an IMDB company that focuses on theatrical releases and their box office performance. This website provided the following box office receipt data:

- Opening weekend US box office dollars
- Opening weekend US screens
- Gross US box office dollars
- Gross US screens

Scraping data from BoxOfficeMojo was a simple case of using the XML package and the readHTMLTable function, to grab the contents of the Nth table from a set of URL's.

The example code below, shows how to grab the 7th table in a HTML page (counting by <table> tag occurrence) and skip the first row of the table (in this case used by a header row that had an inconsistent layout with the rest of the table)

```
  table <- readHTMLTable(url, skip.rows=1,as.data.frame=TRUE)[[7]]
  names(table) <- c("Title", "Studio", "Domestic.Gross",
"Domestic.Theaters", "Opening.Gross", "Opening.Theaters", "Open.Date")
```

However, this approach, whilst simple, is dependent upon all the URL's that will be requested, having a consistent page layout. To prove the point, the structure of the

required web pages changed in mid March (2012) so that the data in question is in table 4, not 7.

For more details, see the file ADempsey_GetBoxOffice.R

Public Opinion

Rotten Tomotoes is a website whose members are able to review and score movies, with smaller number of members deemed 'critics' and their scores and rating provided separately. Rotten Tomatoes therefore, can provide 4 separate measures of popularity, being:

- Critics Rating – 3 categories
- Critics Score – 0 to 100
- Audience Rating – 3 categories
- Audience Score – 0 to 100

The critics and the audience sometimes agree, sometime they do not.

Scraping the Rotten Tomatoes data was done through their JSON API, registration for which is free. Data is available for public non-commercial use, although is limited in terms of the volume of requests per day.

The required packages are RJSONIO for dealing with JSON formatted data and RCurl for handling HTTP requests. Rather than building a set of URL's as was done with BoxOfficeMojo, the Rotten Tomatoes API allows for dynamic queries. Therefore, I used the list of movie names I obtained from BoxOfficeMojo.com, with the API.

```
  rottoms.url <-
paste('http://api.rottentomatoes.com/api/public/v1.0/movies.json?q=',movie.plus,'&year=',movie
[2],'&apikey=',my.key,sep='')

  rottoms.out <-getURLContent(rottoms.url,curl=getCurlHandle()) # this is the data
  doc <- fromJSON(rottoms.out) # return a list structure of the movies that match the title
```

However, the Rotten Tomatoes API returns both exact and partial matches, so filtering of the returned results is essential. Additionally, the structure of the JSON objects can be inconsistent from movie to movie, so even with the 'exact match', care must be taken extracting the details.

```
if(index != 0){
  rat <- which(names(doc[[2]][[index]]) == "ratings")
  cr <- which(names(doc[[2]][[index]][[rat]]) == "critics_rating")
  cs <- which(names(doc[[2]][[index]][[rat]]) == "critics_score")
  ar <- which(names(doc[[2]][[index]][[rat]]) == "audience_rating")
  as <- which(names(doc[[2]][[index]][[rat]]) == "audience_score")
  dtls <- data.frame(title = movie,
                     year = year,
                     critics_rating = ifelse(length(cr) != 0, doc[[2]][[index]][[rat]][cr], NA),
                     critics_score = ifelse(length(cs) != 0, doc[[2]][[index]][[rat]][cs], NA),
                     audience_rating = ifelse(length(ar) != 0, doc[[2]][[index]][[rat]][ar], NA),
                     audience_score = ifelse(length(as) != 0, doc[[2]][[index]][[rat]][as], NA),
                     stringsAsFactors=FALSE)
```

For more details, see the file ADempsey_RottenTomatoes.R

Movie Critics Opinion and Naïve Sentiment

One potential issue with the ratings and scores from RottenTomatoes, is that they could suffer from a varying 'base opinion' over time, as the members contributing as either critics or general audience are not constant. A more consistent source of opinion scores could likely be the New York Times.

The New York Times offers an API that allows for queries to extract movie review information in the form of critics scores and even the original published review article. The API requires a free registration in order to access it, and can provide the following data:

- 1000 best
- Critics pick
- Published review article URL

This is an XML API and as such, the package XML is used again but this time with the function xmlParse in order to extract details from the data returned. As with RottenTomoatoes, I used the list of movie names from BoxOfficeMojo and multiple movies may be returned with each search, so care must be taken to only accept the best match.

I then used the published review article URL to scrape the review text, and performed a naïve sentiment analysis of the article that basically counted the number of negative and positive words used in the review, to produce an overall sentiment score (positive words – negative words = sentiment). This idea came from a winner of the 2012 R in Business competition by Revolution Analytics. As the article in the link states, this analysis is very naïve as it does not take into account the context of the words, nor is it able to detect the use of sarcasm.

Extracting data from the New York Times proved to be the most complex of my sources, as well as the most error prone. For some reason, most likely that of

popularity of the public API, obtaining results for my list of movies took multiple loops through my list. E.g.

- Loop 1 – query for all movies
- Loop 2 – query for all movies without results from loop 1
- Loop 3 – query for all movies without results from loops 1 and 2
- Etc.

See the files ADempsey-GetNYTReviews.R and ADempsey-NYTSentiment.R for more details

Merging Data by Movie Name

The final step in collecting my data was that of merging it together into a single file. Doing so for data from BoxOfficeMojo, RottenTomatoes and the New York Times was a simple join on the movie name (BoxOfficeMojo was used for the list of movies for the other two sources).

Merging with my data for DVD unit sales was not so simple, due to the movie names being different. In order to perform this last merge, I did some manual edits to remove punctuation and then started to look for matches between my datasets.

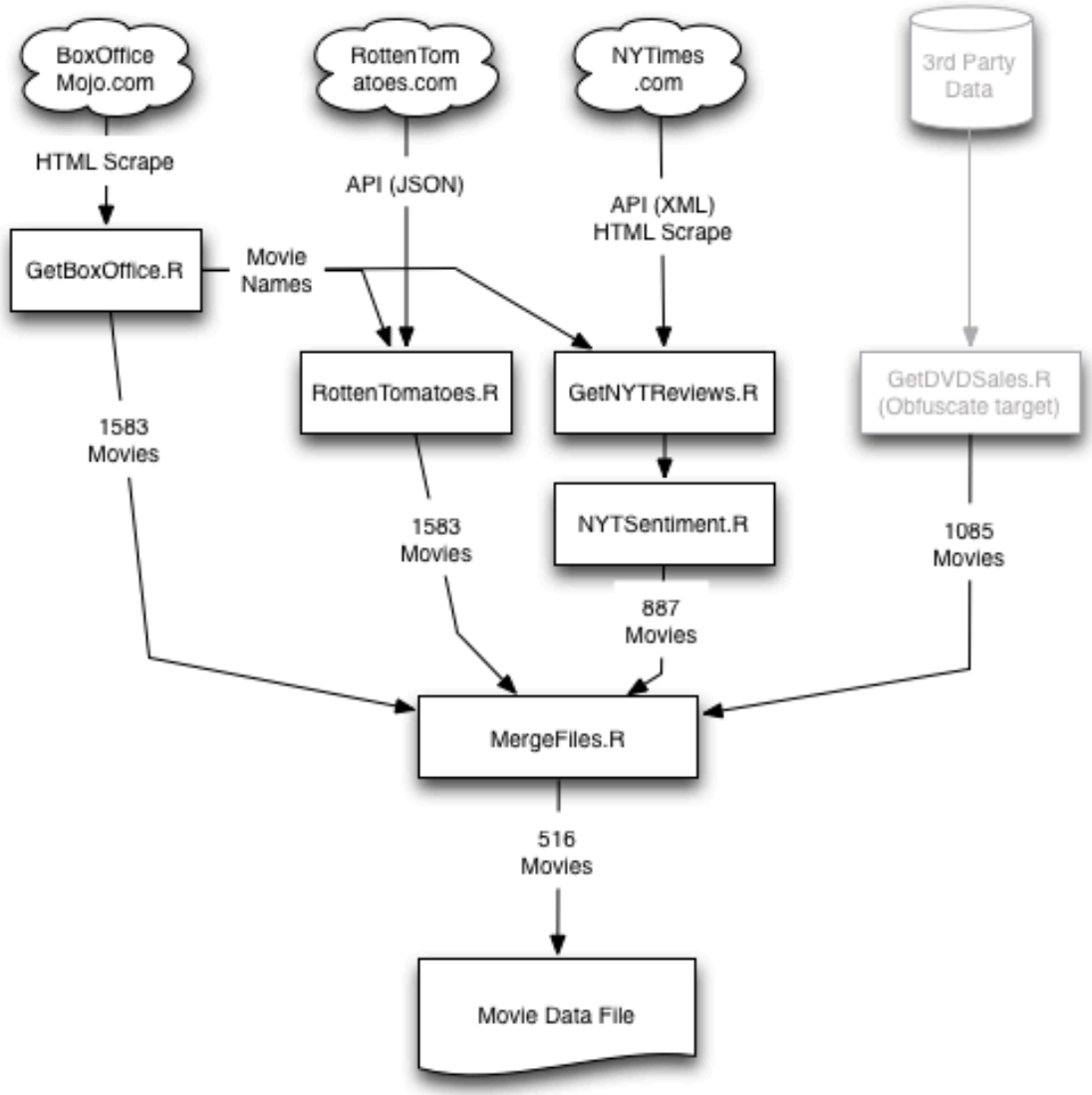
If a clean match did not exist between my scraped data and my DVD sales data, my code then looked for partial matches (e.g. 'A-Team' and 'The A Team'), and output the possible matches to the console in order for me to review and select which one was the appropriate match (if indeed there was one).

One issue I faced with my data was that of missing values in some columns. The approach I took with dealing with these columns was:

- RottenTomatoes Critics and Audience score – linear regression
- RottenTomatoes Critics and Audience rating – assigned based upon where the scores fell in various range buckets
    - Ranges obtained from reviewing complete records to find the break points
- New York Times 1000 best and critics choice assumed to be zero if missing

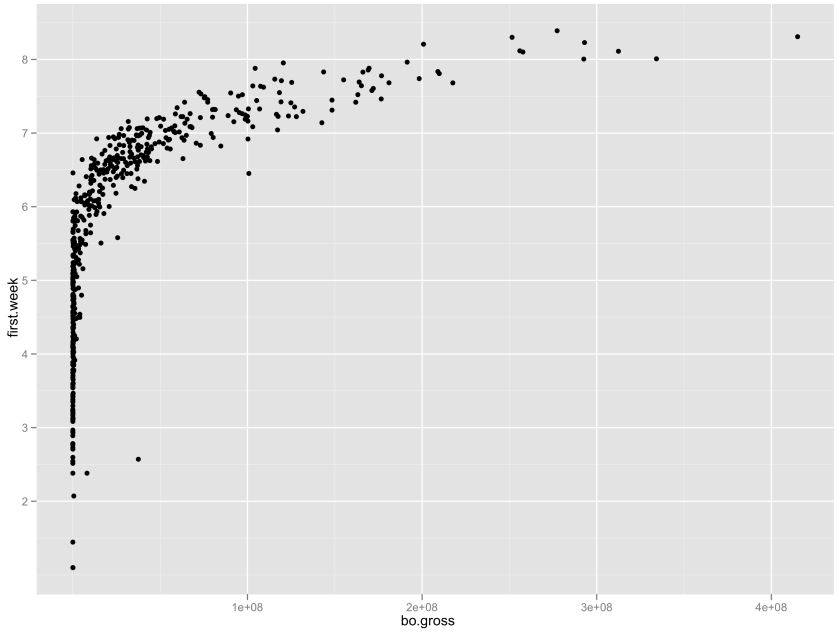
Some of the above approaches are not ideal and can introduce errors into the final modeling. But for the purpose of this project to develop a framework to perform bulk model build and test with, I deemed these issues to be acceptable. Any real world implementation of such a system would obviously require a much more robust data collection and merging mechanism, such as a commercial ETL tool.

As a result of using 4 different data sources, which did not have a consistent set of data for all the movies I originally extracted from BoxOfficeMojo.com, my final data set was about one third of what I stated with.

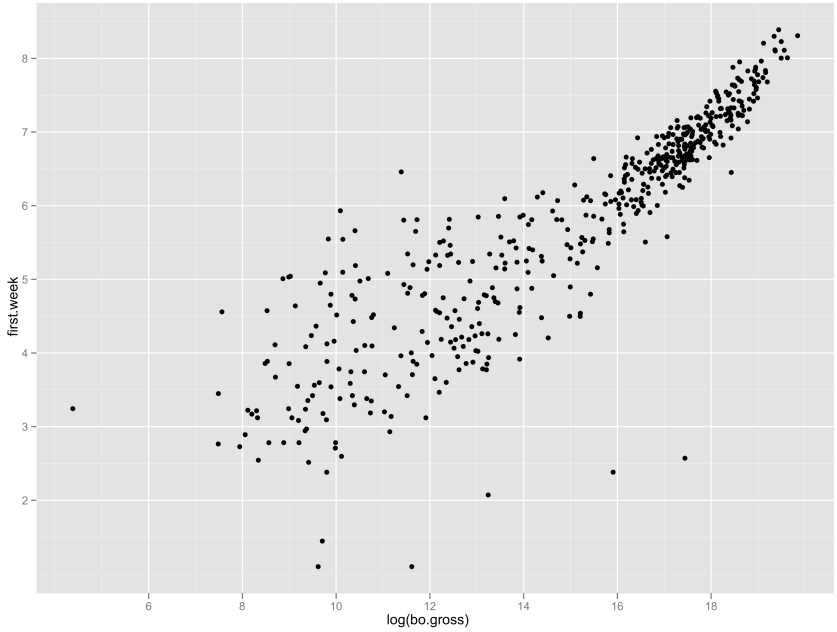

BoxOffice Mojo.com
RottenTom atoes.com
NYTimes .com
3rd Party Data
HTML Scrape
API (JSON)
API (XML) HTML Scrape
GetBoxOffice.R
Movie Names
RottenTomatoes.R
GetNYTReviews.R
GetDVDSales.R (Obfuscate target)
1583 Movies
1583 Movies
NYTSentiment.R
1085 Movies
887 Movies
MergeFiles.R
516 Movies
Movie Data File

Data Analysis

A brief analysis of my data shows a potential relationship exists between box office receipts and DVD sales volume.



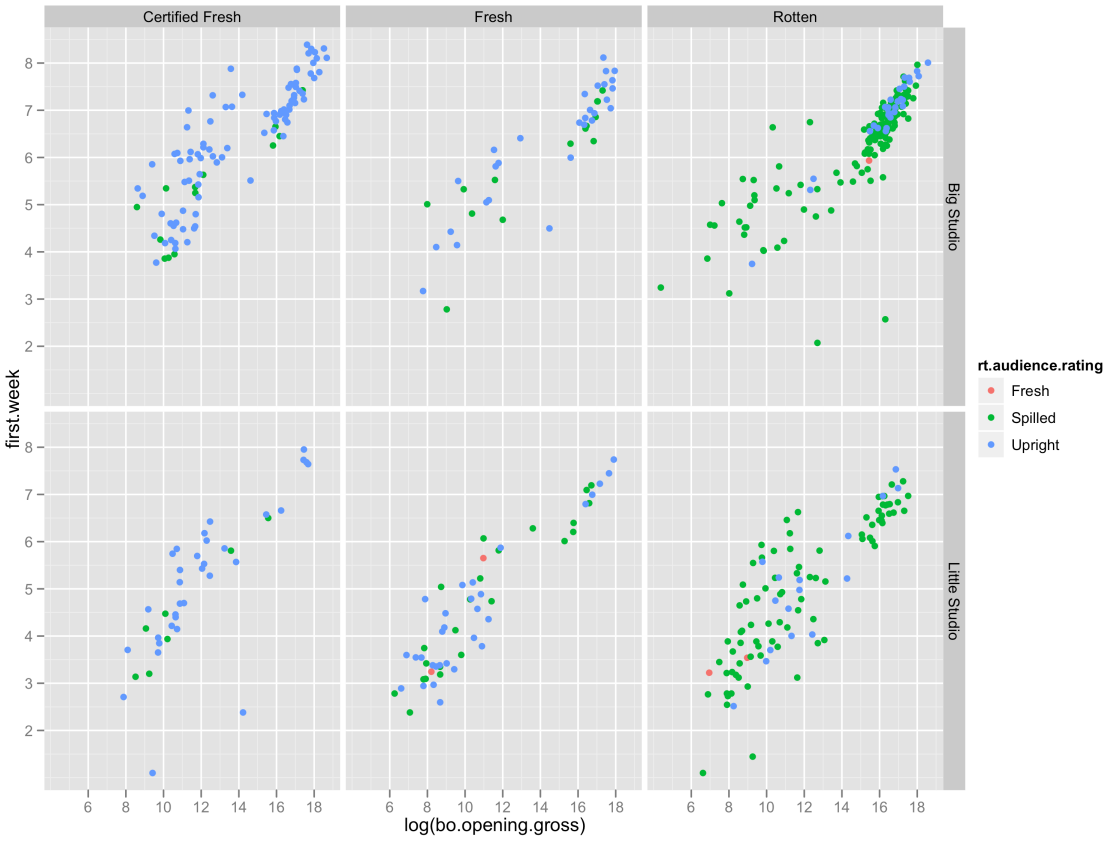
Using the log of box office receipts produces something more linear looking. A log transform to the box office fields should help any resulting model from being overly influenced by the simple magnitude of these numbers, when compared to all other input columns.



I performed two further tweaks to the data file:

- Mapping movies to ‘Big Studios’ (15 or more movies ion the timeframe) and ‘Small Studios’ (12 or lass movies in the timeframe)
- Pivoting out the categorical columns (factors) of ratings, into binary flag columns (numerical)

Reviewing the log of box office receipts with the addition of some categorical columns (RottenTomatoes Critics Rating, Audience Rating and “Big/Little Studio) shows that the potential relationship appears to be broadly applicable.



After all of this, I was left with the following data (including the pivoted columns):

- Not used. For human information only, mostly categorical
  - Movie title
  - Year of theatrical release
  - RottenTomatoes critics rating
  - RottenTomatoes audience rating
  - Studio
  - Studio group
- Numerical input columns

    - RottenTomatoes critics score
    - RottenTomatoes audience score
    - US Box office gross
    - US Box Office screens
    - US Box office opening weekend
    - US Box office opening weekend screens
    - Log US Box office gross
    - Log US Box office screens
    - Log US Box office opening weekend
    - Log US Box office opening weekend screens
    - Open date (days since 1/1/1970)
    - New York Time critics pick
    - New York Times 1000 best
    - New York Times sentiment score
    - Big studio
    - Little studio
    - Critics rotten
    - Critics fresh
    - Critics certified
    - Audience spilled
    - Audience fresh
    - Audience upright
- Target
    - First week DVD sales
    - Frist 4 weeks DVD sales
    - First 8 weeks DVD sales

Model Training and Prediction

I decided to try the following combinations of input columns, with which to model and predict each of the three target columns. In each case, only the numerical input columns are considered.

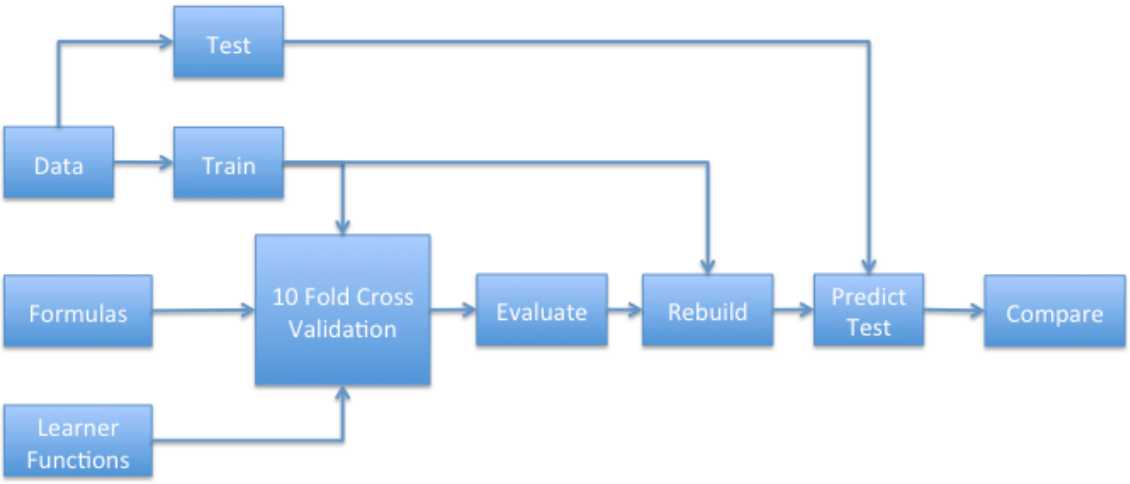
- All input columns
- Early read columns
    - Columns such as those from the New York Times and opening weekend box office receipts, being those columns available before others such as audience ratings. An early read, might give competitive benefits in terms of purchasing the desired volume before supply becomes limited
- 'Best 5 columns'
    - The best 5 columns were determined for each target column, by using a random forest function that looks at the increase in mean squared error when each input column is removed in turn

These three combinations of columns, were done with either the raw box office numbers or the log box office numbers, not both – in other words, 6 combinations of columns were used to predict 3 different target columns - A total of 18 different formulas.

Each of these formulas was then used with 6 different regression models, to give a total of 108 models, not counting for any tuning parameter combinations.

- Decision trees
- Random forest
- Linear regression
- Support vector machines
- Neural networks
- Multiple attribute regression splines

When adding in the different combinations of parameters to these algorithms, a total of 3024 models were built and tested. The process of all these models builds was aided significantly by the use of the framework demonstrated in the book "Data Mining with R – Learning with Case Studies" by Luis Torgo.



This framework, at a high level, uses three key components:

- A list of lists containing formula plus data set
- Custom learning functions that must accept 'formula', 'training' and 'test' data and '…' for additional algorithm tuning parameters, and returns a performance metric (in my case, the mean squared error)
- A cross validation function that uses the above two items and performs all of the necessary splits and executions

After a random split of 100 records into a test data set, this framework (once the bugs had been worked out) allowed all 3024 models to be built and tested on the training data, in 94 minutes.

Prior to this final execution, several initial passes were performed on a subset of formulas (6), using abroad range of function parameters, was performed in order to find an appropriate range for the final execution. These investigational passes took a total of 1000 minutes of execution time and generated several thousand models.

The net result of these initial passes, apart from the appropriate parameters for the final execution, was the exclusion of certain algorithms from the final pass. Both polynomial and radial basis function SVM's were excluded from the final pass, as the best parameters from the broad testing showed that best models were more than likely going to be linear (although sigmoidal SVM's showed some promise when using the shorter, log based formulas). Also, I could not write a ridge regression function that worked with expected pattern of functional call for this framework.

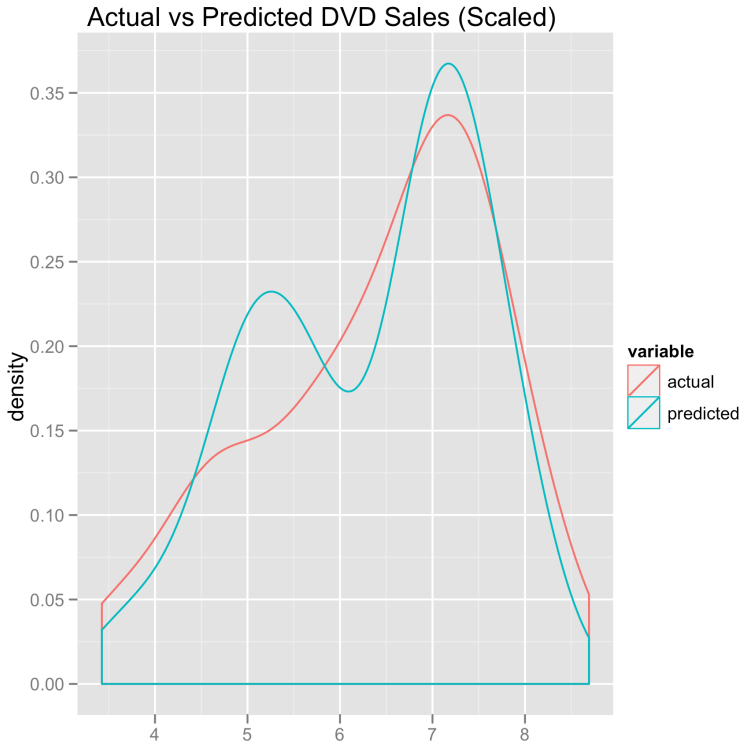
Unfortunately, there were some issues with input data columns containing nothing but zero's in some splits of the cross validation. In these instances, the SVM models were erroring out with a scaling problem. Due to time constraints, I ended up removing both New York Times critics pick and 1000 best flags from some of the formulas, in order to avoid the errors.

Once all of the models had been built, additional functions in the books package, 'DMwR', allow for the identification of the best model for each formula (based on smallest mean squared error), along with the extraction of the parameters used in those formulas. The combination of model algorithm and parameters used was used to rebuild the models using all the training data and evaluate them with the test data. From these final 18 models, the best one was identified by the smallest mean squared error.

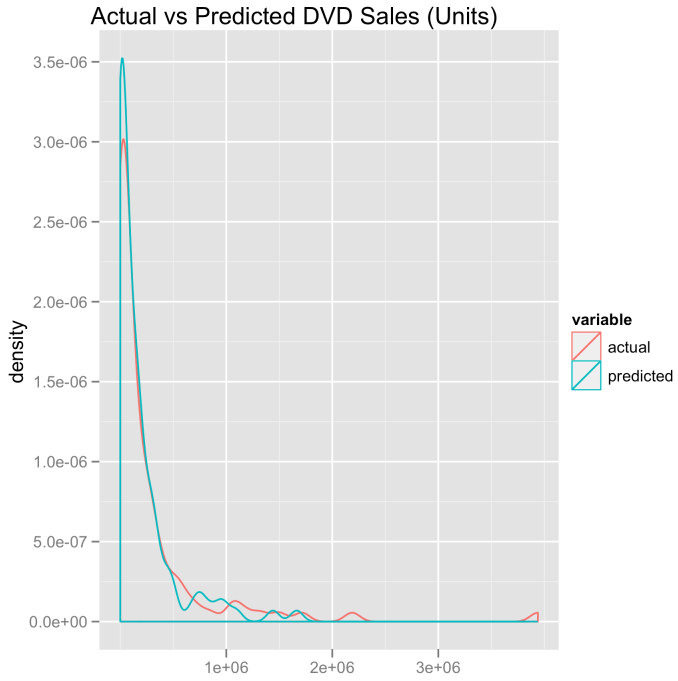
The best model was:

- Formula '12'
  - first.8.weeks ~ rt.critics.score + rt.audience.score + log.bo.gross + log.bo.screens + log.bo.opening.gross + log.bo.opening.screens + open.date + nyt.critics.pick + nyt.sentiment.score + crit.rotten + crit.fresh + crit.certified + aud.spilled + aud.fresh + aud.upright + little.studio + big.studio
  -
- Random Forest
  - nTree = 250
  - maxnodes = 25
  - MSE = 0.24029

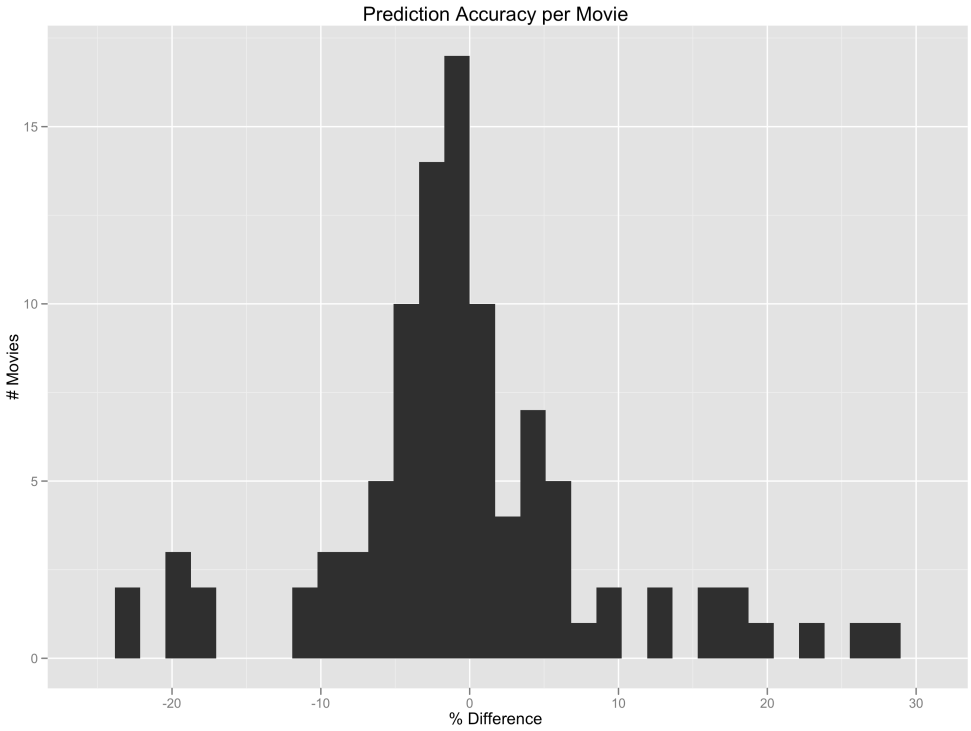
Plotting the performance of this model on the test data, gives the following, showing both over and under prediction.



Transforming the obfuscated target variable back into the real numbers, shows that most of the over prediction occurred for movies with lower sales volumes.



Slicing the data a different way, and looking at the percentage difference in prediction and actual sales volumes, shows that most predictions were within 10% of the actual. Unfortunately, a 10% difference in the real numbers, could be anywhere from 20,000 to 500,000 sales units.



For more details, especially the framework used in the code, please refer to the file ADempsey-BuildAndPredict.R for mode details.

Issues and Improvements

Whilst the overall concept has been proved, that a mix of popularity data can be used to predict DVD retail unit sales, the overall accuracy of this first attempt would be unacceptable for a real world implementation. The following is a list of ways in which performance of predictions should be able to be improved:

- Better data quality through the use of an appropriate ETL tool
    - Better scraping functionality
    - More robust file merging
- Better handling of missing values
    - Perhaps the use of K nearest neighbors instead of a linear regressions
- Additional data sources and columns
    - DVD format – but this would mean even more target columns
    - Genre – Different genres attract different audiences, but also sell different formats to different extents (e.g. 'The Help" likely will not sell Blu ray and 3D in quite the same proportions as "Avatar" will)
    - Series – To a certain extent, subsequent 'Harry Potter' DVD's, benefit from the reputation of previous 'Harry Potter' films
    - 'Successful Director' and 'Popular Hollywood Star' (as of the date of theatrical release and DVD street date) flags
    - Better sentiment analysis that takes into account the context of the usage of any particular word.
    - Additional sentiment sources with a broader range of opinions, rather than a single source

In addition to the above data improvements, there is a potential that the code framework that was used could be parallelized easily using R 2.14 and the 'parallel' package, in order to reduce the execution time. However, the issue with this is in the use of an 'S4' object (the list of lists of formula plus data, I believe) which is not easily vectorized for parallel execution.