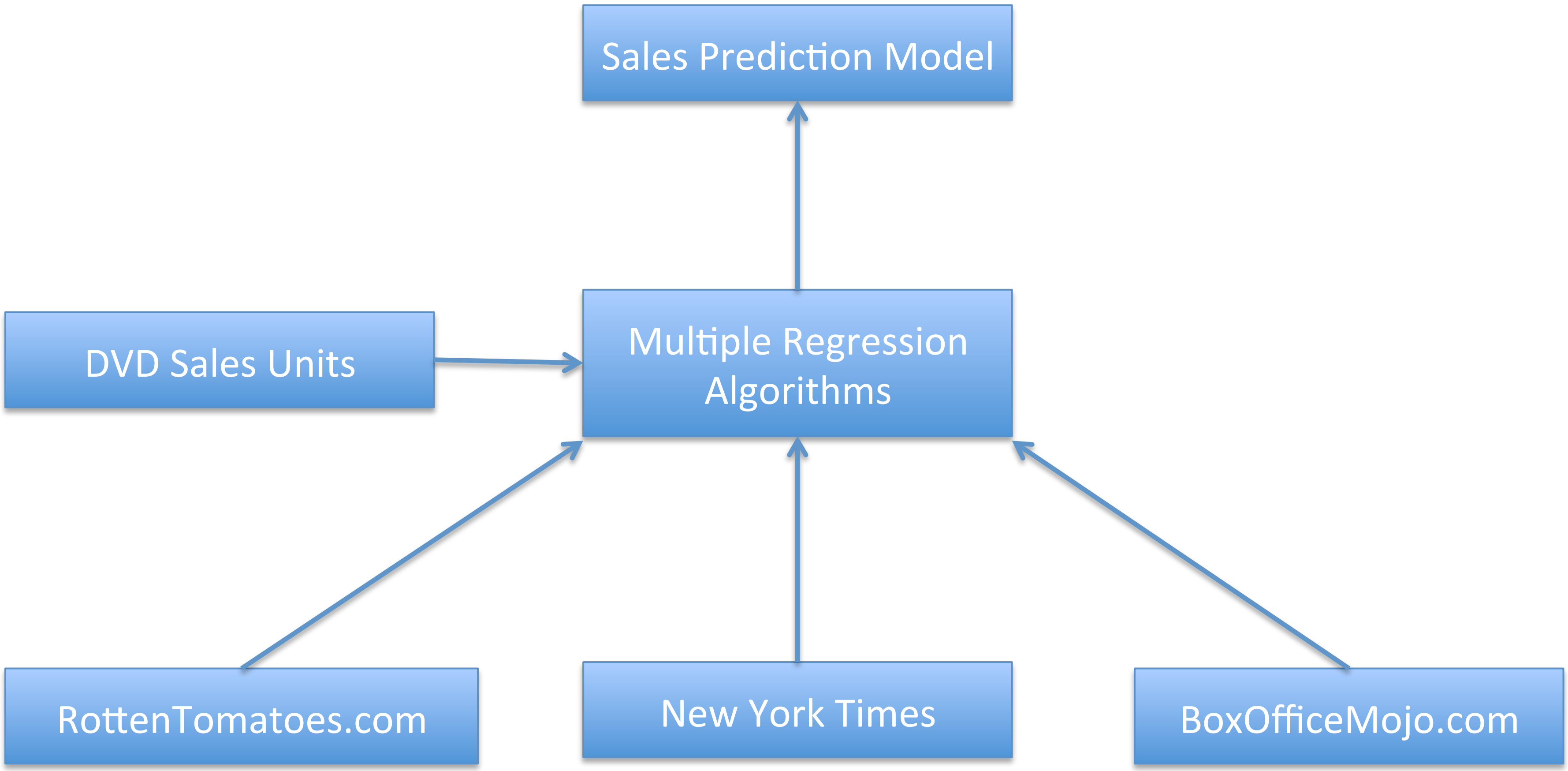
Predicting DVD unit sales based on popularity measures

Andrew Dempsey
Director, BI Architecture
Netflix DVD
UCSC Silicon Valley Extension
2612 Introduction to Machine Learning and Data Mining

Box office receipts are a reliable indicator of DVD unit sales

Additional popularity information, such as reviews, should add more accuracy to any prediction

Basic Intent



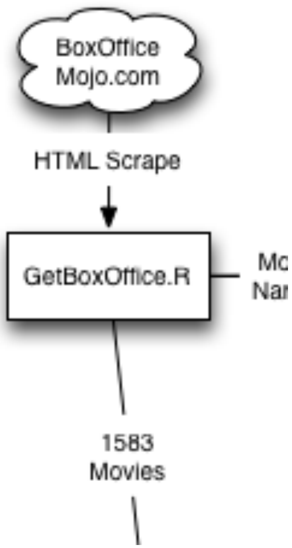
R, with my programming skills, does not make a good ETL tool

GETTING MY DATA

At a High Level

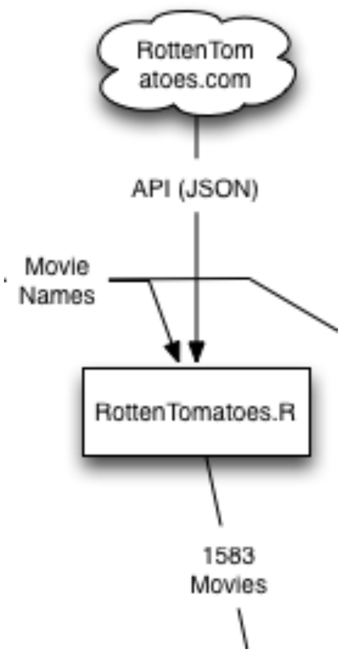
- Scrape data from several websites or extract from their API’s
- Limit data to 2009 – 2011 theatrical release movies
- Merge by movie name

BoxOfficeMojo.com



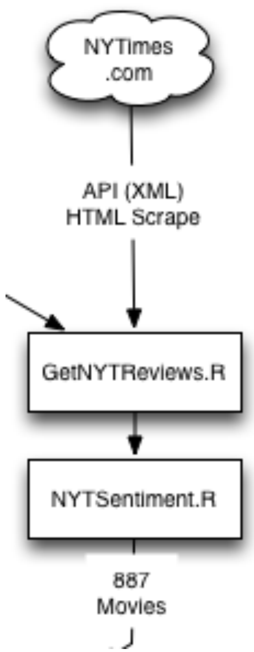
- First week and gross US box office receipts
- Package: XML
- Function: readHTMLTable
- Build array of URL’s
- Scrape each URL and extract Nth <table> to a data frame
- Filter to 2009-2011
- EASY
- Quick example code…

RottenTomatoes.com



- Popular opinion scores and ratings
- Packages: RJSON, RCurl
- Functions: getURLContent, fromJSON
- Use list of movie names from first step
- Query JSON API and only take EXACT MATCH
- Much HARDER

New York Times

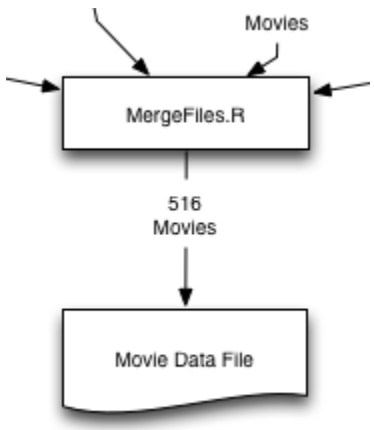


- Major publication critic scores and reviews
- Package: XML
- Function: xmlParse
- Use list of movie names from first step
- Query XML API and only take EXACT MATCH
- Scrape URL given by API to grab review
- Count positive and negative words
- HARD
- Temperamental

Somewhere Else..

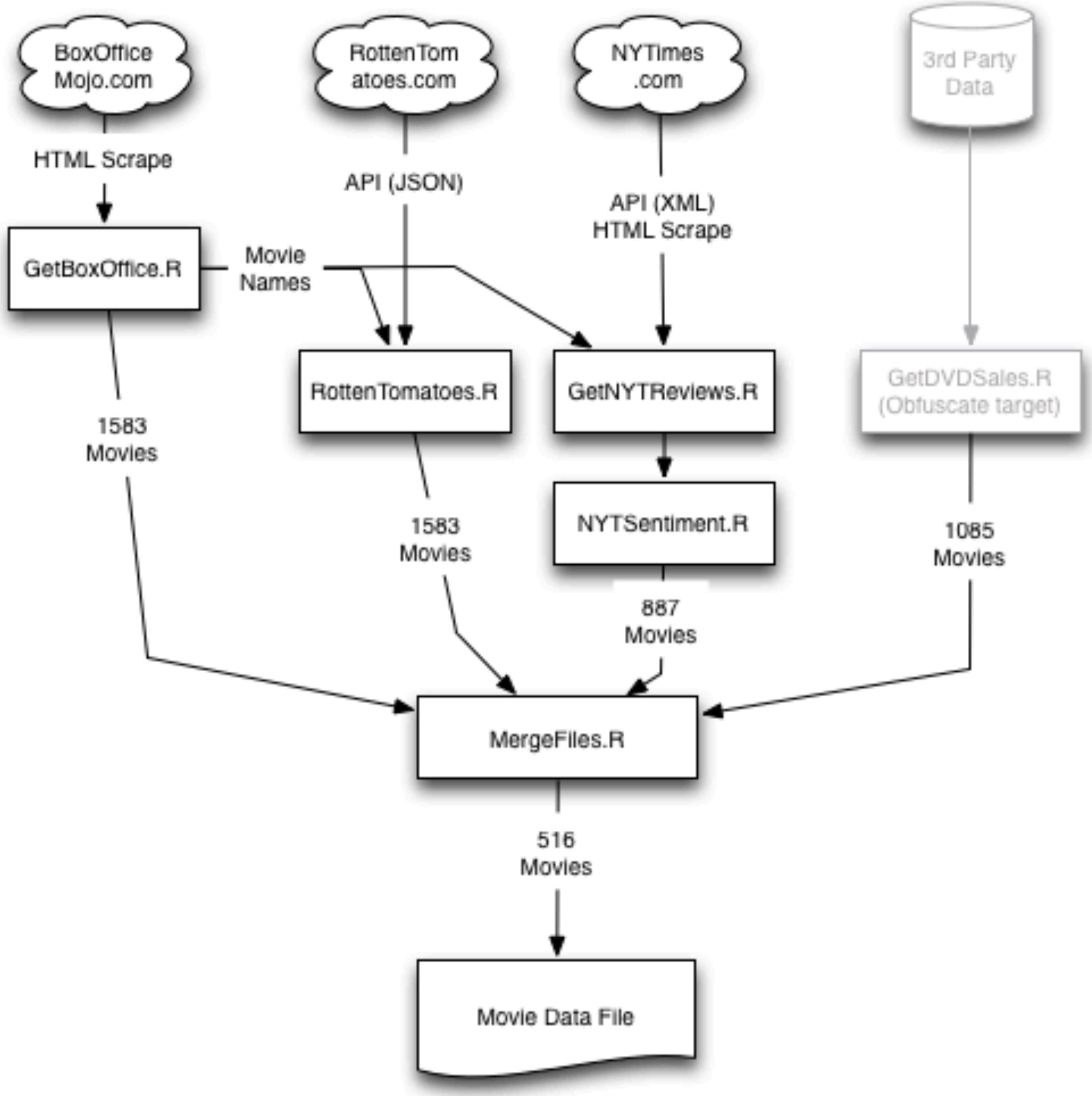
- DVD Sales Units
- Filter to 2009-2011
- Obfuscate sales units

Merge by Name



- Exact match
- Submatches using grep / regex + Manual entry
- Filling in missing values
- Much data loss

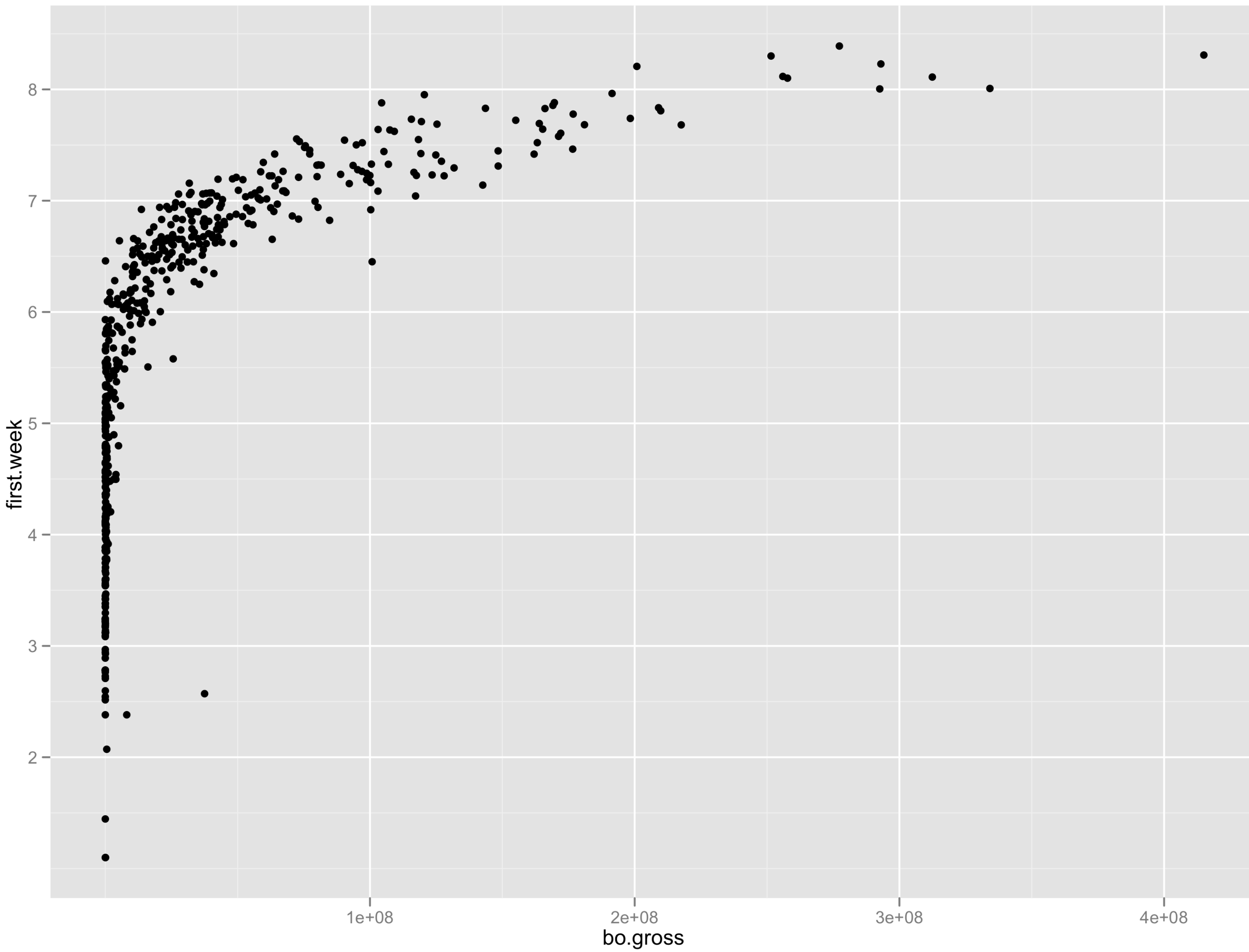
Overall


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
NYTSentiment.R
1583 Movies
1085 Movies
887 Movies
MergeFiles.R
516 Movies
Movie Data File

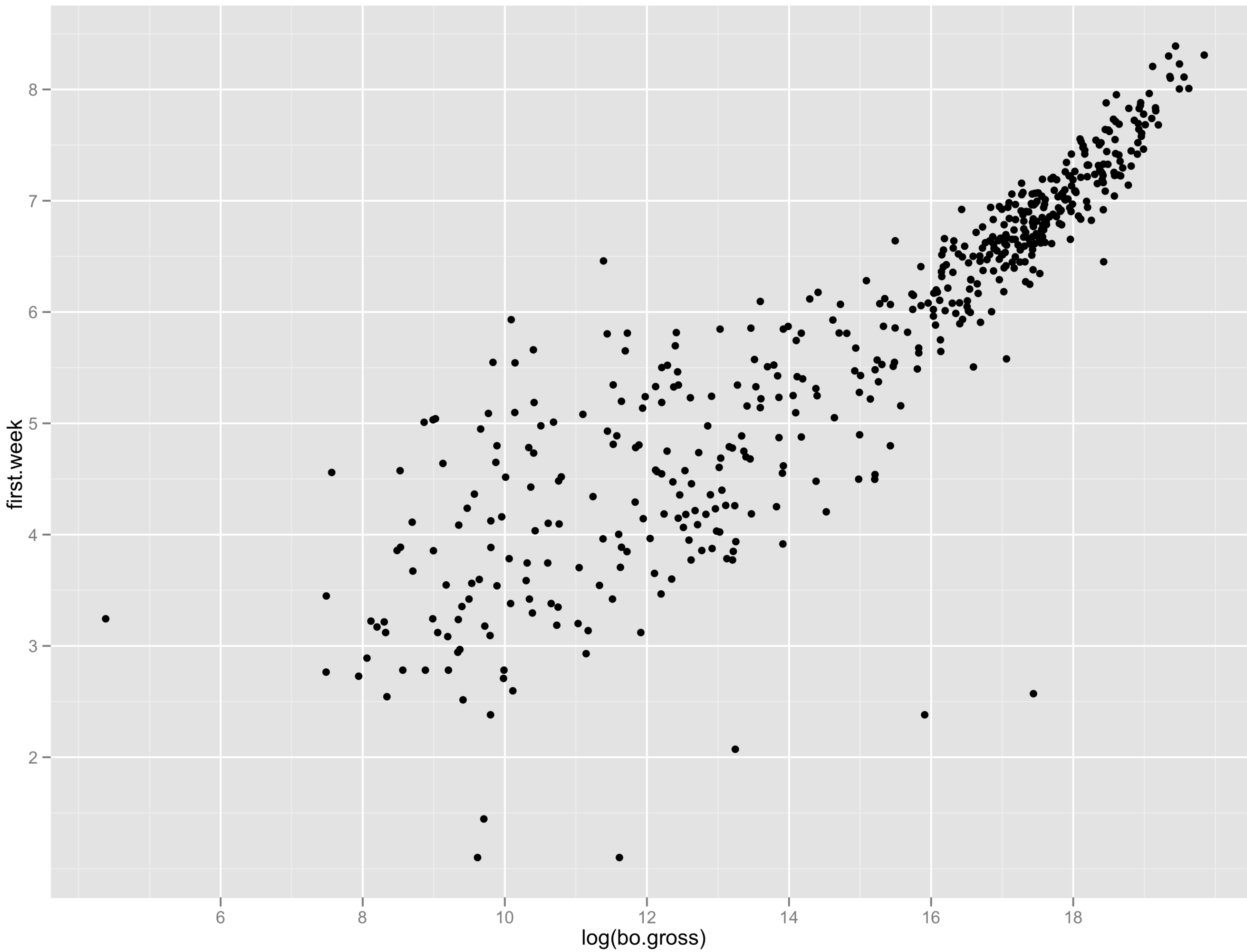
What is my data?

- Inputs
  - Categorical – Rating classification
    - ‘Certified Fresh’
  - Big integer numbers – Box office dollars
    - Hundreds of millions
  - Small integer numbers – Rating scores
    - 0-100
  - Tiny integer numbers – Flags
- Targets
  - 3 Tiny-ish continuous numbers (0-10)
    - Unit sales for first week
    - Unit sales for first 4 weeks
    - Unit sales for first 8 weeks

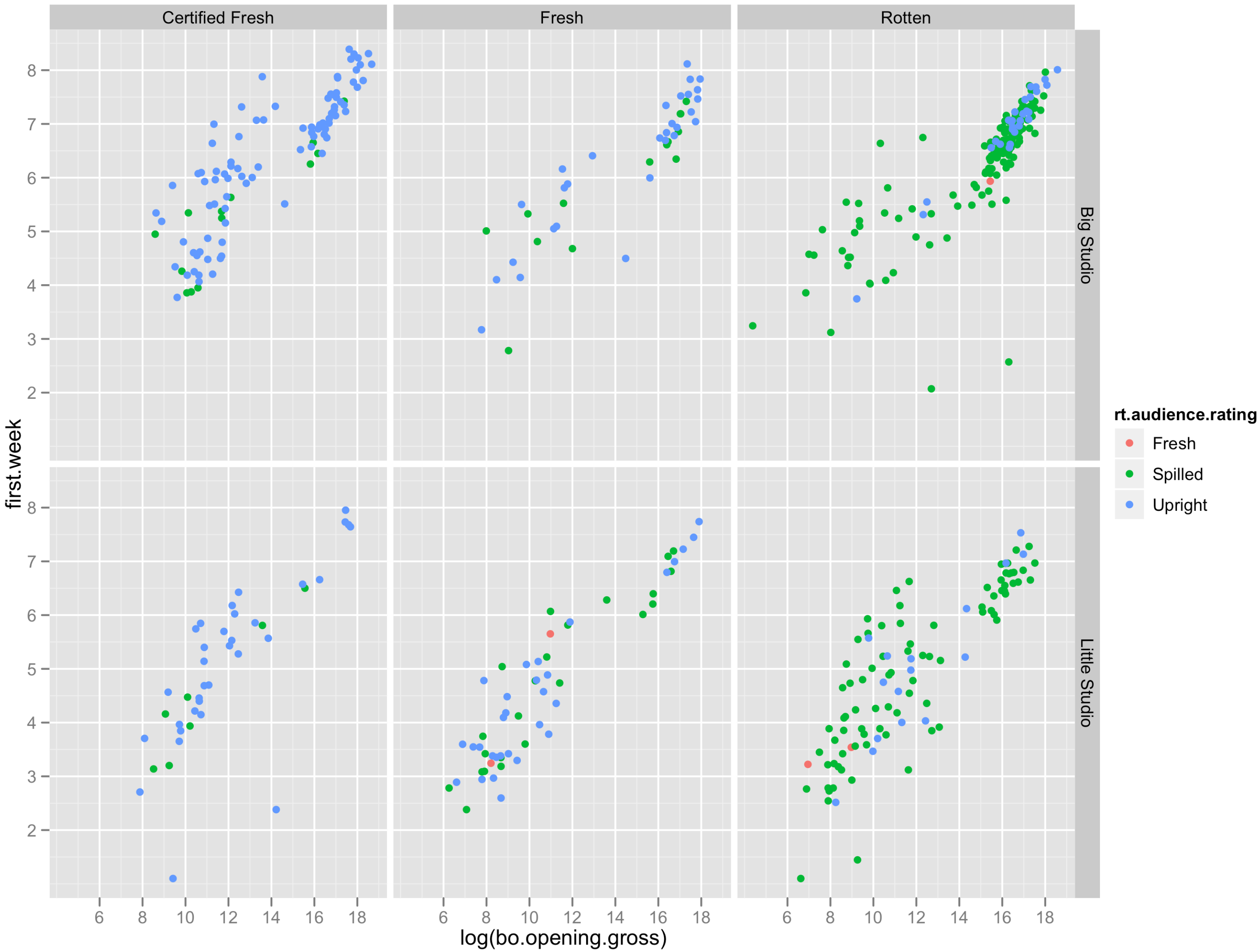
What is my data?



What is my data?



What is my data?



What will I use to predict?

- 3 formulas
  - All columns
  - Early read columns
  - The ‘best 5’ columns
    - randomForest(my.formulas[[1]], data=movie.train,importance=TRUE)$importance
    - Returns the increase in MSE when a column is removed from the model
- 3 targets x 3 formulas = 9 formulas

What inputs will I use to predict?

- 6 formulas
  - All columns
  - Early read columns
  - The ‘best 5’ columns
  - All columns with log box office numbers
  - Early read columns with log box office numbers
  - The ‘best 5’ columns with log box office numbers
- 3 targets x 6 formulas = 18 formulas

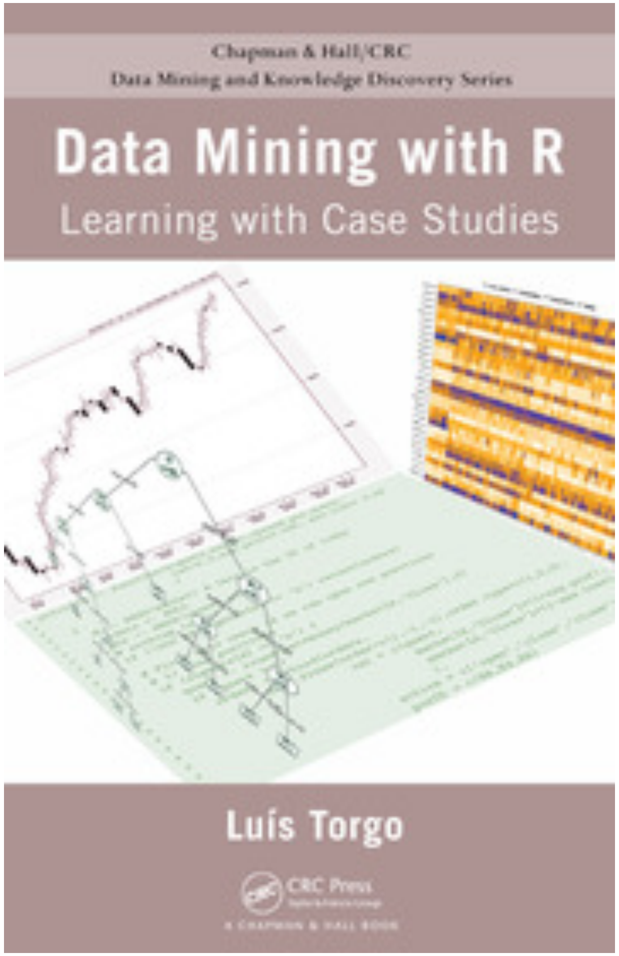
How will I predict

- Decision Trees
- Random Forests
- Linear Regression
- Support Vector Machines
- Neural Networks
- Multiple Attribute Regression Splines
- 6 algorithms x 18 formulas = **108 models**

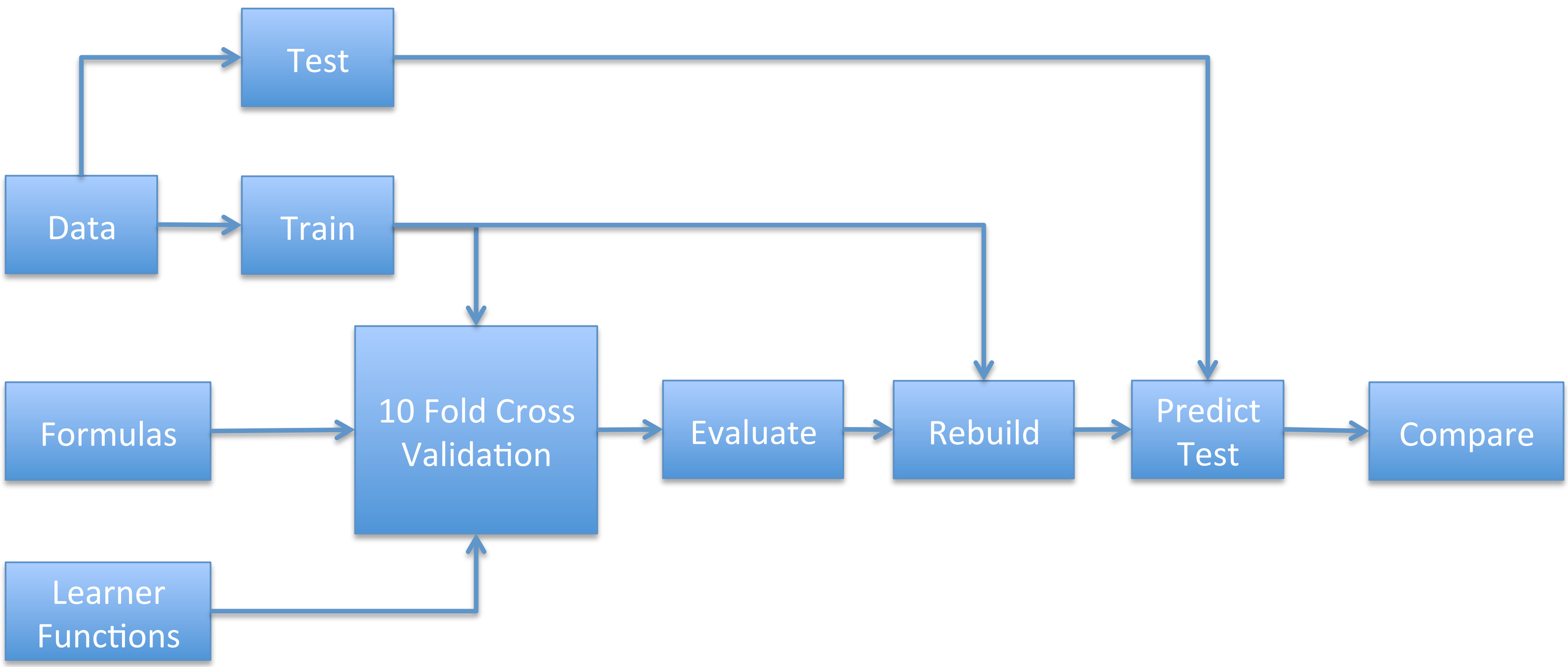
What about all the different function parameters…

THOUSANDS OF COMBINATIONS?

A Book to the Rescue



High Level Program Flow



Formulas

- List of Lists

```
formulas <- (list(
        dataset(target~inputs, dataframe),
        dataset(target~inputs, dataframe),
        dataset(target~inputs, dataframe))
```

Learner Functions

- Simple function pattern

```
my.learner <- function(formula, train, test, …){
  mod <- algorithm(formula, train, …)
  pred <- predict(mod, test)
  mse <- mean((pred - resp(formula, test))^2)
}
```

Bulk Learner Execution

- Built in cross validation

```
res.all <- experimentalComparison(
      formulas,
      c(variants('my.dt', se=c(0,0.25,0.5,1)),  # 4 DT
      variants('my.rf', ntree=c(100,250,500,750,1000),
                  maxnodes=c(5,10,15,20,25,30))  # 30 RF
      ),
      cvSettings(1,10,1234))
```

Which is Best?

- Simple function to find the best

```
best.builds <- sapply(bestScores(res.all),function(x) x['system'])
```

- Returns values like ‘my.rf.v16’

- Simple function to find the parameters used

```
params.used <- lapply(best.builds,function(x)
getVariant(x,res.all)@pars)
```

- Returns values like ‘ntree=75, maxnodes=30’

Testing the Best Models

- Rebuild the model

```
best.models[[a]] <- do.call(funcs.used[[a]], c(list(my.formulas[[a]],
movie.train), params.used[[a]]))
```

- Apply to test data

```
test.preds[i,] <- sapply(1:length(best.models), function(x)
predict(best.models[[x]],movie.test[i,]))
```

- Which is best?

```
best.models.details[[which(test.preds.mse == min(test.preds.mse))
[1]]]
```

Finding Good Parameters

- Run each algorithm individually, with a broad parameter set
- 1000 minutes of parameter tuning!

```
res.all <- experimentalComparison(
  test.groups,
  c(
    # Inital testing of linear (51 variations - 30 mins) gave best performance at cost = 0
    variants('my.linsvm',kernel='linear', cost=seq(0,10,0.2)),
    # Inital testing of polynomial (4410 variations - 379 mins), gave best performance at cost=0, gamma=0, degree=1, coef0=0
    variants('my.polysvm',kernel='polynomial', cost=seq(0,10,0.5), gamma=seq(0,10,0.5), degree=(1,2), coef0=seq(0,2,0.5)),
    # Inital testing of radial (441 variations - 33 mins), gave best performance at cost=0, gamma=0
    variants('my.rbfsvm',kernel='radial', cost=seq(0,10,0.5), gamma=seq(0,10,0.5)),
    # Inital testing of sigmoid (4410 variations- 412 mins), gave best performance at cost=0, gamma=0, degree=1, coef0=0
    variants('my.sigsvm',kernel='sigmoid', cost=seq(0,10,0.5), gamma=seq(0,10,0.5), degree=c(1,2), coef0=seq(0,2,0.5)),
    # Initial testing of ANN (300 variations = 138 mins), give best values maxit=(100,200,300,400,600), decay=(0.01, 0.1), size=(2,8,10,16,18)
    variants('my.nnet', linout=TRUE, maxit=seq(100,1000,100), size=seq(2,20,2), decay=c(0.001,0.01,0.1), entropy=FALSE)#, # 300 ANN - 138 mins
    ),
  cvSettings(1,10,1234))
```

Execution

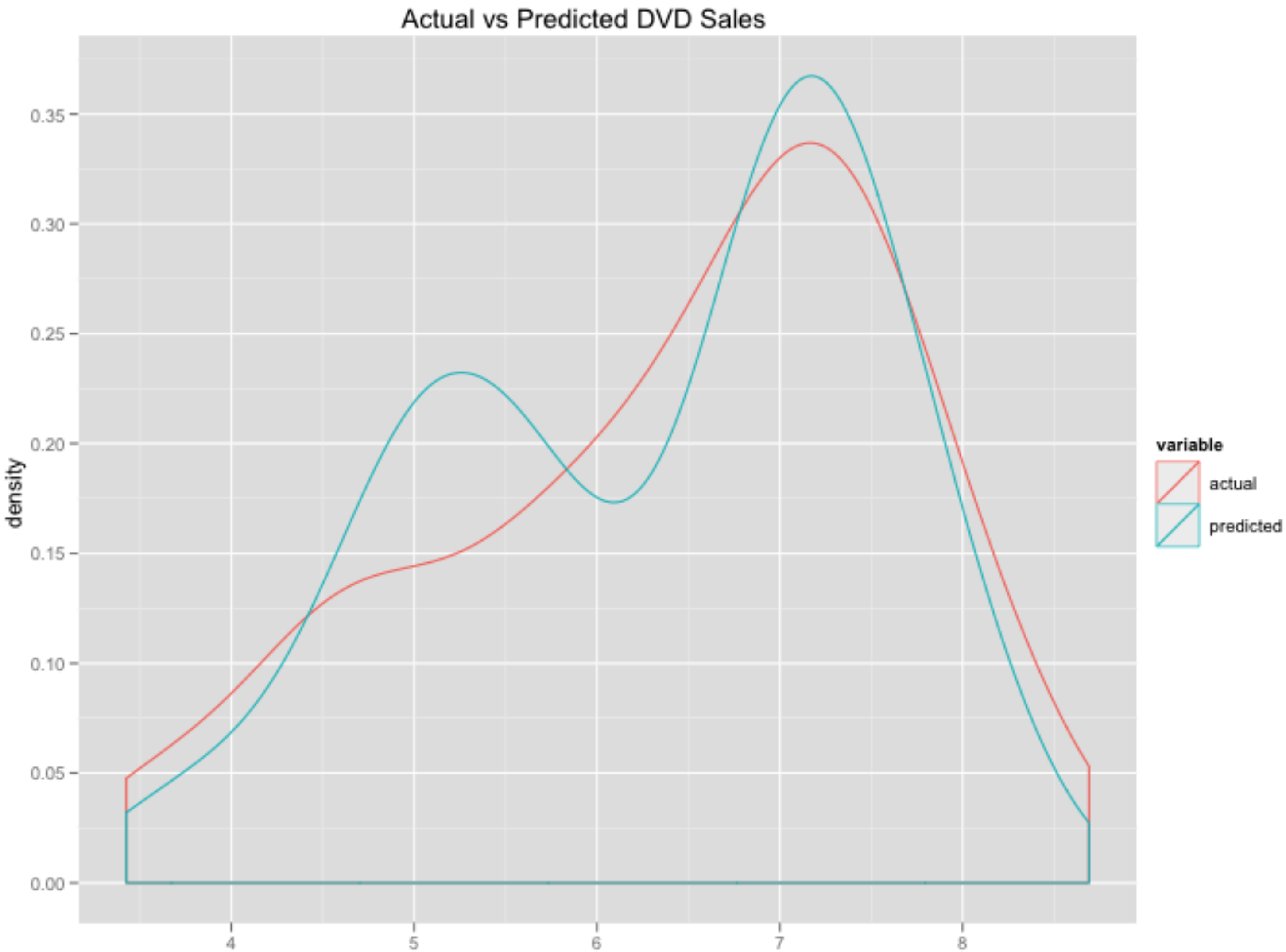
- 164 models for 18 formulas = 3024 models
  - 94 minutes

```r
res.all <- experimentalComparison(
  test.groups,
   c(variants('my.dt', se=c(0,0.25,0.5,1)),  # 4 DT
     variants('my.rf', ntree=c(100,250,500,750,1000), maxnodes=c(5,10,15,20,25,30)),  # 30 RF
     variants('my.lm'),  # 1 LM - returns "prediction from a rank-deficient fit may be misleading" errors when using the 'best columns' formulas with 6 columns?
     variants('my.linsvm',kernel='linear', cost=seq(0,1,0.05)), # 20 lin-SVM
     variants('my.sigsvm',kernel='sigmoid', cost=seq(0,0.2,0.05), gamma=seq(0,0.2,0.05), degree=1, coef0=seq(0,0.5,0.5)), # 50 sig-SVM
     variants('my.nnet', linout=TRUE, maxit=c(100,200,300,400,600), size=c(2,8,10,16,18), decay=c(0.01,0.1), entropy=FALSE), # 50 ANN
     variants('my.mars', nk=c(3,4,5), degree=c(1,2), thresh=c(0.0001,0.001,0.01)) # 24 mars - keep running into subscript out of bounds and other errors when using the 'best columns' formulas with 6 columns?
    ),
  cvSettings(1,10,1234))
```

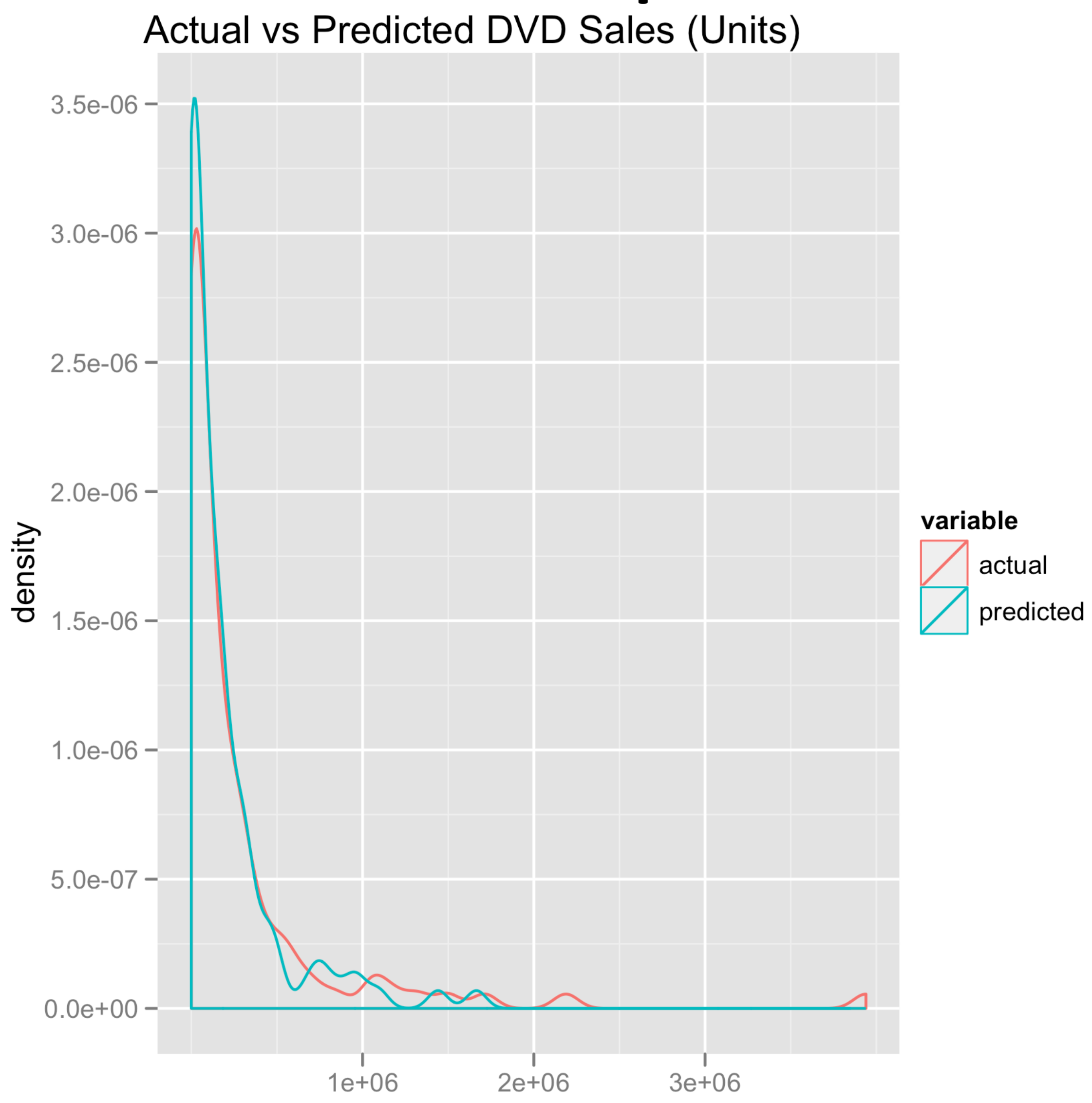
Which model was best?

- Best
  - 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 = 0.2402864 (scaled)

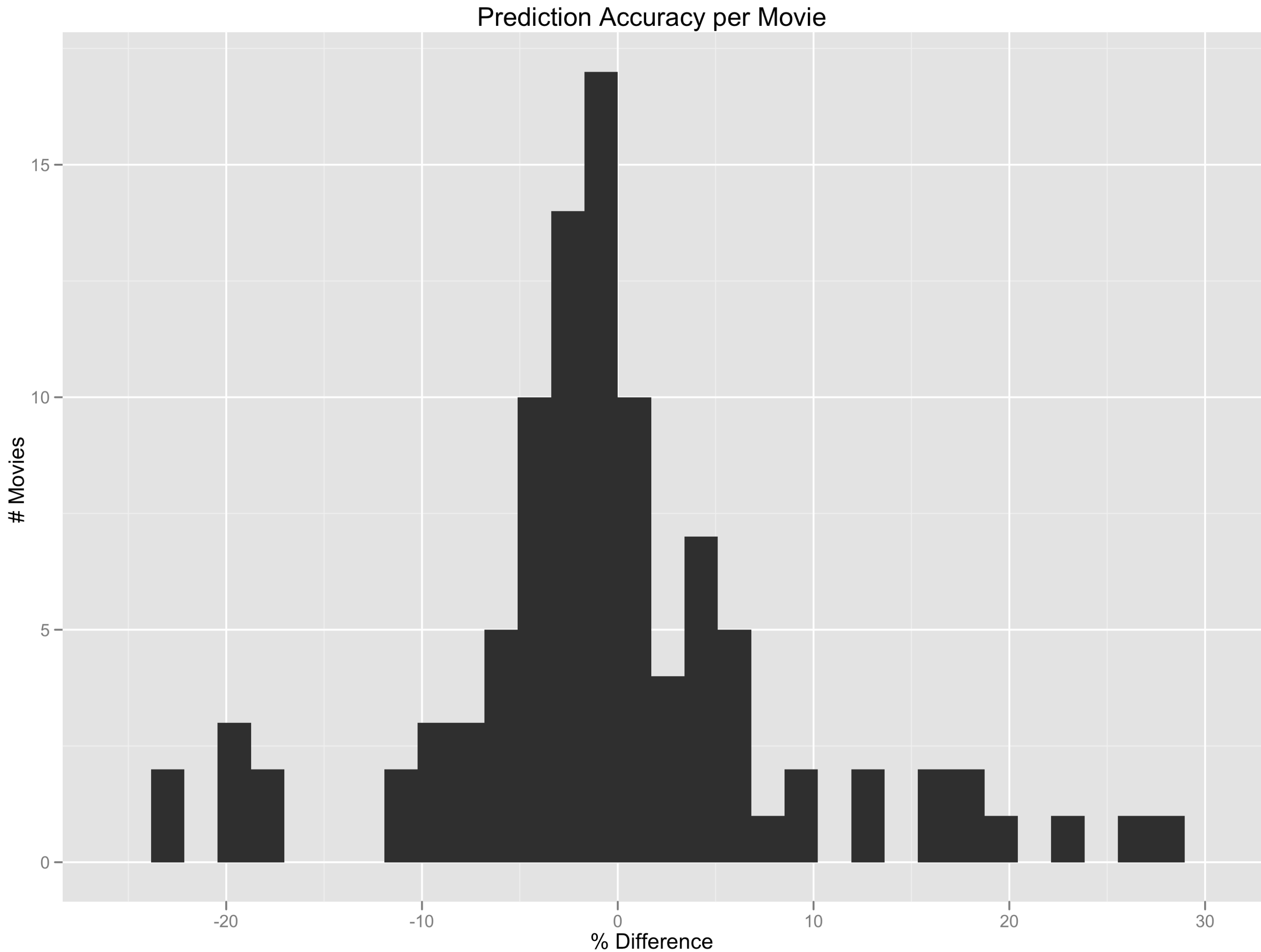
How did it perform?



How did it perform?



How did it perform?



Problems I encountered

- StackOverflow.com helped with some scraping and parsing code
- Merging multiple files by movie name
  – Manual edits
- Missing values
  – I did a quick linear regression to fill in some missing values
- Some input columns were very sparse and caused scaling errors in SVM function
  – So I dropped them!

Improvements?

- IMPROVE DATA QUALITY
  - Use a real ETL tool
- Parallelize the ExperimentalComparison function
  - R 2.14 has an in built parallel package
- Less hardcoding of formula strings
- More inputs
  - 'Genre'
  - 'Series' (e.g. 'Harry Potter')
  - Flags for 'successful director', 'current major Hollywood star' etc.
  - Better sentiment analysis